Universidad de las Ciencias Informáticas

Facultad 3

Título: Herramienta para la evaluación cuantitativa de la confiabilidad de la arquitectura de Cedrux

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor:

Yandy González Cañizarez

Tutor(es):

Msc. Larisa González Alvarez

La Habana

Junio 2013

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Yandy González Cañizarez

_____ _____

Msc. Larisa González Alvares (Co-Tutor)

Datos de contacto

AGRADECIMIENTOS

A mi tutora por haberme guiado con tanta paciencia y dedicación durante toda mi tesis.

A mis padres, hermanos y a toda mi familia.

A mi las personas que siempre los he querido como padres y hermanos, a Frank y Cristina, Leonel, Duniel.

A mis amigos y hermanos de la universidad Angel Alexander (Alex), Alfredo, Carlos, David, Hernan, Luis Angel (el mono), Luis Angel (el calvo), Cesar, Diogenes, Raidel, Jiubel, Rogelio Veitia(el bolo), Pompa, Ramon.

A mis compañeros de aula, los que estuvieron y están desde primer año hasta 5to.

A mis amigas: mis …… Liliana, Danay, Lili.

Siempre los tendré en mi corazón, aunque yo sea pequeño mi corazón tiene espacio para todos.

DEDICATORIA

Dedico mi tesis en especial a mi hermana Yelena.

A mis padres Esmirtha y Jesús por ser mis guías en todo momento, por inculcarme siempre buenos valores, por siempre estar ahí cuando lo he necesitado.

A mi hermano Yoandy por apoyarme siempre.

A mi sobrino que lo quiero con el alma.

RESUMEN

La confiabilidad es un atributo de calidad relevante para los Sistemas de Planeación de Recursos Empresariales. Por tal motivo, el presente trabajo tiene como objetivo la implementación de una herramienta informática para el análisis cuantitativo de la confiabilidad en la arquitectura del sistema Cedrux, que contribuya a la obtención de datos precisos tras la aplicación del Modelo de estimación de la confiabilidad de los componentes de software mediante el aprovechamiento de los modelos arquitectónicos (MECCMA). Para su cumplimiento se realizó un estudio de las herramientas, los lenguajes y la metodología de desarrollo empleada utilizándose como IDE de desarrollo el Netbeans 7.1.2, PostgreSQL 8.3 como gestor de base datos, para el diseño nos apoyamos en la herramienta Visual Paradigm 8.0 y el lenguaje Java para la implementación de la aplicación, todo esto basado en la utilización de la metodología Extreme Programming (XP) . Se presentan los artefactos generados durante el proceso de desarrollo de la herramienta, y al finalizar se muestran las pruebas efectuadas para detectar y corregir los errores antes de la entrega al cliente. Con la implementación de la herramienta informática antes mencionada, se facilitará a los arquitectos del sistema Cedrux, medir la confiabilidad a los componentes que existen en el mismo, arrojando como resultados datos cuantitativos que serán tenidos en cuenta en el momento de la toma de decisiones en la evaluación arquitectónica, todo esto dependiendo de los errores arrojados por cada componente.

Palabras Claves: Confiabilidad, evaluación arquitectónica, modelo matemático, componente.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los sistemas de Planificación de Recursos Empresariales (ERP[1], por sus siglas en inglés) son sistemas de información integrados, compuestos por un conjunto de módulos funcionales susceptibles de ser adaptados a las necesidades de cada cliente (Ricart, 2009). Un sistema ERP combina la funcionalidad de los distintos programas de gestión en uno solo, basándose en una única base de datos centralizada. De esta manera permite garantizar la integridad y unicidad de los datos a los que accede cada departamento, evitando que tengan que volver a ser introducidos en cada aplicación o módulo funcional que los requiera.

La mayoría de los ERP, adopta una estructura modular que soporta los diferentes procesos de una empresa: el módulo de gestión financiera, el módulo de gestión de compras, el módulo de logística, el módulo de recursos humanos, entre otros (Gómez Vieites, 2011). El hecho de que estos productos sean modulares posibilita la implantación del sistema por etapas. El propósito fundamental de un ERP es otorgar apoyo a los clientes del negocio, tiempos rápidos de respuesta a sus problemas, así como un eficiente manejo de información que permita la toma oportuna de decisiones y disminución de los costos totales de operación.

La Universidad de las Ciencias Informáticas (UCI) y específicamente el Centro de Informatización de la Gestión de Entidades (CEIGE) cuenta con una línea de producción de software para el entorno empresarial, este tiene como principal objetivo el desarrollo de una serie de sistemas que brindarán una solución nacional que permita, a diferencia de otros productos similares, la gestión integral de las entidades presupuestadas y empresariales, basadas en los principios de independencia tecnológica con funcionalidades generales de los procesos y las particularidades de la economía cubana (Toro, 2007).

Una de las etapas fundamentales en el proceso de desarrollo de un ERP, se ve reflejada en la construcción de la arquitectura. No pocos son los autores que plantean que el éxito o fracaso de un sistema está determinado por esta (Gómez, 2007) .La arquitectura afecta a todo lo relacionado con el proyecto: afecta a los clientes, al gerente, al equipo de desarrollo, y al equipo de pruebas. Cada interesado se preocupa por partes específicas del sistema, y esto se ve reflejado en la arquitectura.

[1] Enterprise Resources Planning

INTRODUCCIÓN

La arquitectura provee una vista de alto nivel de cómo las funcionalidades y responsabilidades del sistema fueron particionadas y luego asignadas a subsistemas o componentes. Su objetivo principal es obtener una comprensión general de cómo y por qué el sistema fue dividido en partes y cómo esas partes interactúan juntas para lograr la funcionalidad deseada (Codorníu, 2010).

Si la arquitectura se encuentra deficiente en su concepto o diseño existirá una gran probabilidad de que el sistema no sea confiable y que no cumpla con las necesidades de los usuarios finales. Por lo que existe la necesidad de una evaluación cuantitativa que refleje resultados que permitan agilizar el proceso de desarrollo. El equipo de arquitectura de Cedrux, trabaja en pro de la identificación y puesta en práctica de distintas técnicas y métodos para cada atributo en específico y, en este sentido, se han dado avances en cuanto al atributo confiabilidad.

En el año 2012 se realizó un estudio de los diferentes modelos matemáticos existentes para la estimación de la confiabilidad, cuyo resultado fue la definición de un Modelo de estimación de la confiabilidad de los componentes de software mediante el aprovechamiento de los modelos arquitectónicos, adaptado al contexto de desarrollo de la arquitectura de Cedrux**.** La evaluación de la confiabilidad mediante este modelo puede brindar una visión del estado de dicho atributo de calidad en la arquitectura y a su vez retribuir el proceso de desarrollo de software (Pérez, 2012)

MECCMA cumple que (Pérez, 2012):

- Es flexible a modificaciones, presentando módulos enchufables a la hora de definir los cálculos.
- Está basado en el análisis de errores o fallos que son tomados como entrada.
- Permite determinar los errores de mayor peso que afectan el sistema.
- Nos brinda la visión del error en aras de facilitar el proceso de corrección contribuyendo a agilizar el proceso de desarrollo del sistema.
- Es completo y robusto, sin la necesidad de datos resultantes de evaluaciones anteriores, ya que posibilita al arquitecto establecer comparaciones cuantitativas de las confiabilidades de los errores del componente, con el fin de determinar los errores criticos y aquellos que afectan en mayor grado al componente, además permite establecer un criterio del estado de la confiabilidad del componente a partir de la cantidad de errores y de la criticidad de sus confiabilidades.

Sin embargo, los elementos que componen el basamento del Modelo y las modificaciones propuestas en (Pérez, 2012), requieren procesamientos no triviales y en algunos casos voluminosos. Ello se puede evidenciar si se tiene en cuenta que:

- La cantidad de cálculos a realizar aplicando MECCMA estaría condicionado por:
    1. La cantidad de componentes de Cedrux
       La composición estructural interna de componentes de Cedrux está basada en dos niveles (Fernández, 2011): componentes nivel 1 (CN1) y componentes nivel 2 (CN2).
    2. La cantidad de errores que presente cada componente (PE)
       Depende de la iteración de prueba.
    3. Cantidad de cálculos realizados en el modelo
       Serían dos cálculos fundamentales: en la Fase 2 con el cálculo de la confiabilidad del error ante el componente y en la Fase 3 con el cálculo de la confiabilidad del componente.

Si se utilizan los datos de la arquitectura de Cedrux 1.0 plasmados en su expediente de proyecto y en (Pérez, 2012), y tomando los tres puntos anteriores como variables de una ecuación, la cantidad de operaciones de cálculo realizadas (CC) podría determinarse mediante:

CC = (CN1 + CN2) * PE* 2

CC = (90 + 10)*3*2 = 600

Obteniéndose un total de 600 cálculos realizados manualmente.

- La determinación de la confiabilidad de un componente mediante la utilización de MECCMA, plantea la utilización del algoritmo de agrupamiento de las K- medias (MacQueen, 1967). La complejidad de aplicación de este algoritmo se evidencia en su basamento, pues el mismo busca formar clusters[2] de forma recursiva teniendo como datos de entrada las confiabilidades obtenidas a

[2] Grupos usados para representar un conjunto de valores, entre todos los iniciales que tiene algo en común, y se pueden agrupar en función de determinado rango.

partir de los errores detectados en el componente. En cada llamada se calculan los centroides[3] de cada grupo, para finalmente obtener la confiabilidad del componente.

Por el razonamiento anterior queda evidenciado que la realización de dichos cálculos de manera manual afectaría la precisión de los datos obtenidos ya que:

- Se pueden introducir errores, fundamentalmente a la hora de la asignación del peso del error, el cálculo de la confiabilidad del error ante el componente y el cálculo de la confiabilidad del componente.
- El proceso de evaluación sería tedioso y no confiable a los involucrados (arquitectos, evaluadores), debido a los voluminosos y no triviales cálculos que se requieren.

Por lo expuesto anteriormente surge el siguiente **problema a resolver**: La aplicación manual del modelo cuantitativo para la evaluación de la confiabilidad arquitectónica de Cedrux, afecta la precisión de los datos obtenidos. Se plantea como **objeto de estudio**: Modelos para la evaluación cuantitativa de la confiabilidad, enmarcado en el **campo de acción**: Desarrollo de herramientas para la evaluación cuantitativa de la confiabilidad. Para dar solución a la problemática planteada se establece como **objetivo general**: Desarrollar una herramienta que soporte el modelo cuantitativo para la evaluación de la confiabilidad de la arquitectura de Cedrux, favoreciendo así la precisión de los datos obtenidos.

El objetivo general se desglosa en los siguientes **objetivos específicos**:

- Realizar el marco teórico de la investigación para fundamentar los métodos y herramientas de soporte a la solución.
- Desarrollar una herramienta informática para el análisis cuantitativo de la confiabilidad de la arquitectura de Cedrux.
- Validar la herramienta implementada.

Para darle cumplimiento a los objetivos específicos se trazaron las siguientes **tareas de investigación**:

- Realizar el marco teórico de la investigación.
- Estudiar las metodologías y herramientas para el desarrollo de software.
- Identificar los requisitos funcionales y no funcionales del sistema.

[3] Es el punto promedio de todos los puntos de un grupo.

- Realizar el diseño de la aplicación y validación del mismo.
- Implementar la aplicación a partir de los diagramas de diseño confeccionados.
- Probar el correcto funcionamiento de la aplicación realizando las pruebas correspondientes

Se define como **Idea a defender**: El desarrollo de una aplicación que soporte el modelo cuantitativo para la evaluación de la confiabilidad en Cedrux, favorecerá la precisión de los datos obtenidos.

Para la realización de la investigación se aplicaron los métodos científicos siguientes:

Del nivel teórico:

- **Análisis y síntesis**: Para el procesamiento de la información y arribar a las conclusiones de la investigación.
- **Histórico-Lógico**: Para determinar las tendencias actuales de desarrollo de los modelos y enfoques arquitectónicos.

Del nivel matemático estadístico:

- **Medición**: Para la obtención del valor numérico de la confiabilidad del componente.
- **Experimental**: Para verificar la precisión de los datos obtenidos mediante el MECCMA.

Este trabajo está estructurado en tres capítulos:

En el Capítulo 1. Fundamentación teórica de la investigación

Comprende la fundamentación teórica de la investigación, la cual ofrecerá información referente a los principales conceptos tratados. Se muestra la metodología, las herramientas y lenguajes seleccionados para el desarrollo de la herramienta.

En el Capítulo 2. Diseño e Implementación del sistema

Se hace énfasis en las características del sistema a desarrollar, se definen las iteraciones a realizar y la duración que presentarán, así como las historias de usuarios y la estimación de esfuerzo que presentan las mismas. Además, se describen las Tarjetas de Cargo o Clase, Responsabilidad y Colaboración (CRC), así como la estructura que presentará la base de datos a utilizar.

En el Capítulo 3. Validación de la solución propuesta

En este capítulo se validará la solución propuesta, mediante la verificación de la precisión de los datos obtenidos por la métrica de precisión propuesta por la norma ISO 9126. En el mismo se evidencia dos métodos de validación: la comparación entre datos esperados y datos obtenidos y la utilización de la métrica.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En aras de establecer un dominio teórico de los elementos fundamentales que soportan la propuesta de solución, se relacionan en este capítulo los principales conceptos asociados a la evaluación de arquitecturas, modelos matemáticos para la estimación de la confiabilidad, las herramientas; incluyendo las de evaluación cuantitativas y las tecnologías para el desarrollo.

1.1 Evaluación de la arquitectura de software

La evaluación de una arquitectura de software surge de la estrecha relación que tiene con la calidad del sistema. De igual manera, serán de particular importancia las propiedades no funcionales del sistema de software, pues influyen notoriamente en la calidad del mismo. Estas propiedades tienen un gran impacto en el desarrollo y mantenimiento del sistema, su operatividad y el uso que éste haga de los recursos (Buschmann, 1996).

Según (Gómez, 2007) “El propósito de realizar evaluaciones a la arquitectura, es para analizar e identificar riesgos potenciales en su estructura y sus propiedades, que puedan afectar al sistema de software resultante, verificar que los requerimientos no funcionales estén presentes en la arquitectura, así como determinar en qué grado se satisfacen los atributos de calidad.”

Evaluar la arquitectura de un software es un medio para la obtención de atributos de calidad del producto que bien pueden servir para:

- Determinar si el software que se está construyendo respalda los no funcionales que le fueron atribuidos.
- Decidir la mejor opción entre arquitecturas candidatas.
- Decidir cuál software, de varios posibles, respalda mejor los requisitos de un cliente.

Evaluar una arquitectura de software es realizar un estudio de factibilidad que pretende detectar posibles riesgos, así como buscar recomendaciones para contenerlos. El propósito de realizar evaluaciones a la arquitectura, es para analizar e identificar riesgos potenciales en su estructura y sus propiedades, que puedan afectar al sistema de software resultante; verificar que los requisitos no funcionales estén presentes en la arquitectura, así como determinar en qué grado se satisfacen los atributos de calidad.

Un atributo de calidad es una característica que afecta a un elemento. Donde el término “característica” se refiere a aspectos no funcionales y el término “elemento” a componente (Ecured, 2008).

Según (Kazman, 2001) el primer paso para la evaluación de una arquitectura es conocer qué es lo que se quiere evaluar, de esta forma, es posible establecer la base para la evaluación, puesto que la intención es saber qué se puede evaluar y qué no. Resulta interesante el estudio de la evaluación de una arquitectura: si las decisiones que se toman sobre la misma determinan los atributos de calidad del sistema, es posible evaluar las decisiones de tipo arquitectónico con respecto al impacto sobre estos atributos.

Es importante tener en cuenta que realizar una evaluación de la arquitectura permite analizar y evaluar la calidad exigida por los usuarios, así como las decisiones del diseño. A grandes rasgos entre los principales beneficios de realizar una evaluación arquitectónica se encuentran: (Carrascoso, 2009)

- Los financieros.
- Los que reúnen a los interesados.
- Las fuerzas de una articulación en las metas específicas de calidad.
- Las fuerzas de una mejora a la documentación de la arquitectura.
- Las mejoras de la arquitectura.
- La detección temprana de problemas.
- Los que validan los requisitos y los priorizan.
- Los que recolectan los fundamentos y las decisiones arquitectónicas tomadas.

Cuanto más temprano se encuentre un problema en un proyecto de software, existirán mayores posibilidades de corregirlo y menores serán los gastos en el proceso. A la vez resulta fácil entender la gran importancia de la evaluación arquitectónica para el buen funcionamiento y la máxima calidad del sistema que se construye. Sin embargo, para que los objetivos de esta sean efectivos, no se puede ignorar la relevancia del momento en que se decida llevar a cabo el proceso.

1.1.1 Técnicas de evaluación

En el campo de evaluación arquitectónica existen determinadas técnicas de evaluación las cuales son instrumentos o herramientas, que conjugadas con otros elementos, proveen a los involucrados en el proceso de una vía para alcanzar sus objetivos y al mismo tiempo pronosticar el comportamiento de la arquitectura de software en su etapa de diseño (Milán, 2011).

Estas técnicas se clasifican en cualitativas y cuantitativas, se utilizan para la evaluación de los atributos de calidad y requieren información del sistema a desarrollar que no está disponible durante el diseño

arquitectónico, sino al principio del diseño detallado del sistema. En vista de que el interés es tomar decisiones de tipo arquitectónico en las fases tempranas del desarrollo, son necesarias técnicas que requieran poca información detallada y puedan conducir a resultados relativamente precisos.

Principalmente las técnicas de evaluación cualitativas son utilizadas cuando la arquitectura está en construcción. Se aplican para la comparación entre arquitecturas candidatas, y tiene relación con la intención de saber la opción que se adapta mejor a cierto atributo de calidad. Donde nos brindan respuestas afirmativas o negativas sin un mayor nivel de detalles. La figura 1 muestra las diferentes técnicas de evaluación.

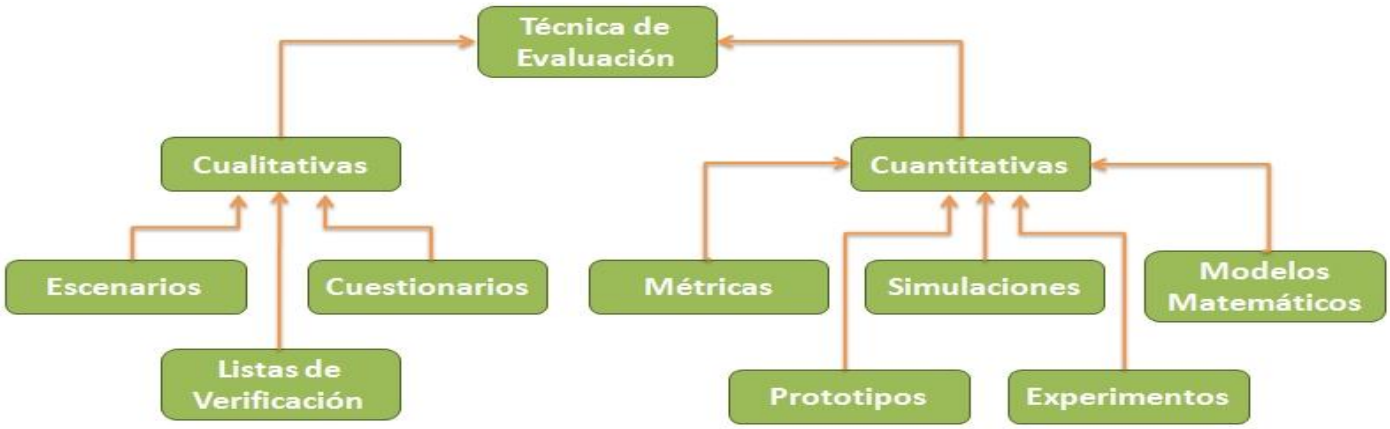

Figura 1. Clasificación de las técnicas de evaluación (Gómez, 2007)

Las técnicas de evaluación cuantitativa se aplican cuando la arquitectura ya ha sido implantada, en busca de la obtención de valores que permitan tomar decisiones en cuanto a los atributos de calidad de una arquitectura de software, comparándolos con los requisitos establecidos y así poder establecer el grado de cumplimiento de una arquitectura candidata, o tomar decisiones sobre ella, pero se ve limitado hasta tanto no se conozcan los valores teóricos máximos y mínimos de las mediciones con las que se realiza la comparación.

1.2 Modelos matemáticos para evaluación de la confiabilidad

La técnica del modelamiento matemático se enfoca en variables como cómputos[4] de alto desempeño (J. Stafford, 1997), sistemas confiables y sistemas en tiempo real, entre otros, donde modelos matemáticos pueden ser usados para evaluar específicamente atributos de calidad. A diferencia de las otras técnicas, los modelos matemáticos permiten la evaluación estática del diseño arquitectural (ERIKA CAMACHO, 2004).

El proceso de evaluación basado en modelos matemáticos sigue los siguientes pasos (Bosch, 2000):

- **Selección y adaptación del modelo matemático:** La mayoría de los centros de investigación orientados a atributos de calidad, han desarrollado modelos matemáticos para medir sus atributos. Estos tienden a ser muy elaborados y detallados, así como también requieren de cierto tipo de datos y análisis. Parte de estos datos requeridos no están disponibles a nivel de arquitectura, y la técnica requiere mucho esfuerzo para la evaluación arquitectónica, por lo que el arquitecto de software se ve obligado a adaptar el modelo.
- **Representación de la arquitectura en términos del modelo:** El modelo matemático seleccionado y adaptado no asume necesariamente que el sistema que intenta modelar consiste de componentes y conexiones. Por lo tanto, la arquitectura necesita ser representada en términos del modelo.
- **Estimación de los datos de entrada requeridos:** El modelo matemático aun cuando ha sido adaptado, requiere datos de entrada que no están incluidos en la definición básica de la arquitectura. Es necesario estimar y deducir estos datos de la especificación de requisitos y de la arquitectura diseñada.
- **Predicción de atributos de calidad:** Una vez que la arquitectura es expresada en términos del modelo y se encuentran disponibles todos los datos de entrada requeridos, el arquitecto está en capacidad de calcular la predicción resultante del atributo de calidad evaluado.

[4] Determinación indirecta de una cantidad mediante el cálculo de ciertos datos. Diccionario Enciclopédico Vox 1. © 2009 Larousse Editorial, S.L.

Se definió la utilización de esta técnica cuantitativa para la evaluación de la confiabilidad de la arquitectura de Cedrux. Obteniéndose como resultado, la utilización del:

Modelo de estimación de la confiabilidad de los componentes de software mediante el aprovechamiento de los modelos arquitectónicos.

Este modelo planteado por (Roshandel, 2006) divide el modelado de un componente en cuatro perspectivas funcionales:

1. La interfaz: Muestra sus servicios prestados y los que se requieren.
2. Comportamiento estático: Muestra la funcionalidad del componente discreto (es decir, en diferentes "Instantes" durante la ejecución del sistema).
3. Comportamiento dinámico: Visión continúa de los detalles de ejecución internos del componente.
4. Interacción del protocolo: Muestra una vista externa continua de un componente en ejecución, especificando la ejecución, permite las huellas de sus operaciones (accede a través de las interfaces).

Lo anterior es utilizado por este modelo de dos maneras importantes. En primer lugar, se utiliza el modelo de comportamiento dinámico como base para el marco de estimación en ausencia de un perfil operacional del componente. En segundo lugar, utiliza el nivel de Interfaz, para mediante la inconsistencia entre los diferentes modelos arquitectónicos, obtener los errores arquitectónicos (o defectos).

Plantea además la construcción de un modelo de Markov de interacción entre los estados individuales de un componente, con el fin de estimar la confiabilidad del mismo. Sin embargo, la falta de un perfil de funcionamiento, hace resultar parámetros desconocidos en el mencionado modelo de Markov. Se tiene por consiguiente, que un Modelo Oculto de Markov (HMM[5]), es un formalismo que puede estimar parámetros ocultos o desconocidos en un modelo, y por lo tanto, este se adecua perfectamente cuando se está en presencia de la falta de un perfil operacional. El marco de estimación de la confiabilidad consta de tres fases (figura 2).

[5] Hidden Markov Model.

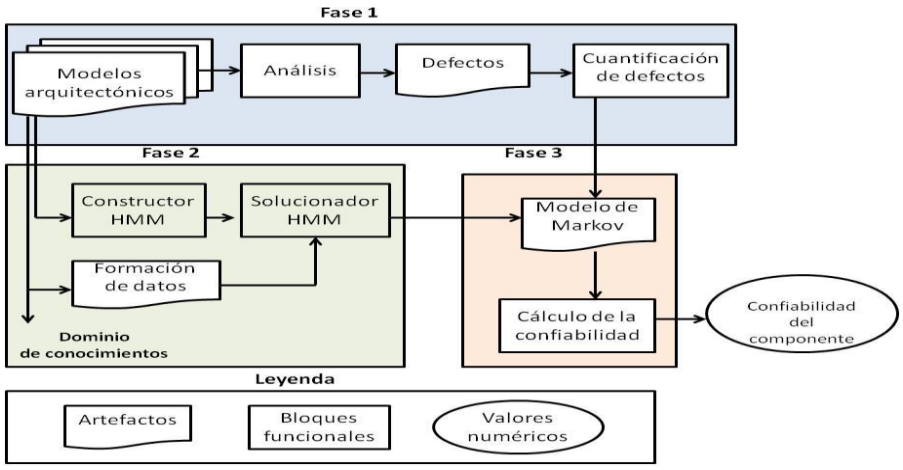

Figura 2. Fases del marco de estimación de la confiabilidad (Roshandel, 2006)

Según (Bosch, 2000) para llevar a cabo un proceso de evaluación mediante modelos matemáticos el primer paso consiste en la selección y adaptación del modelo.

Después de haber seleccionado el Modelo de estimación de la confiabilidad de los componentes de software mediante el aprovechamiento de los modelos arquitectónicos, y además identificadas las principales ventajas y desventajas de este para Cedrux, se realizó la adaptación del modelo. La misma tuvo como primicia mantener las buenas prácticas, así como adecuar los diferentes factores que no se adaptan al sistema.

Fase1. Modelado de la arquitectura, análisis y cuantificación

Para el modelo de evaluación se requiere la utilización de dos vistas.

1. Interfaz: Está definida en Cedrux como de Integración.
2. Comportamiento dinámico: Tiene como homóloga en el sistema Cedrux la Vista de sistema.

En esta fase además se lleva a cabo un análisis en busca de la detección y cuantificación de errores que contenga el componente a evaluar. Teniendo en cuenta el conocimiento de los arquitectos se decidió tomar como referencia sus criterios a la hora de adaptar la clasificación de los errores en correspondencia con el sistema, también a la hora de establecer los costos de los errores según su clasificación, en dependencia de lo dañino que puedan ser. Esta fase se muestra en la Figura 3.

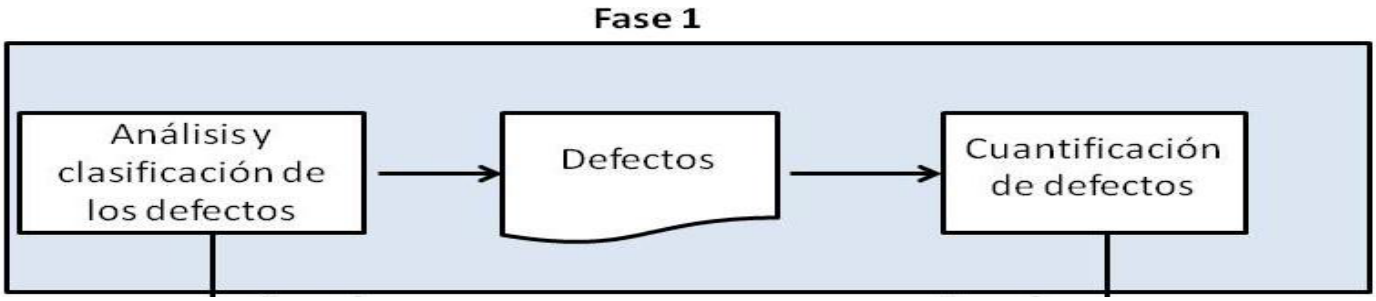

Figura 3. Fase1 del MECCMA (Roshandel, 2006)

Fase 2. Modelado del perfil operacional

Teniendo en cuenta que la formación y generación de datos son a la vez módulos enchufables, se ha decidido la realización de importantes transformaciones que dividen la segunda fase en un módulo principal (Figura 4).

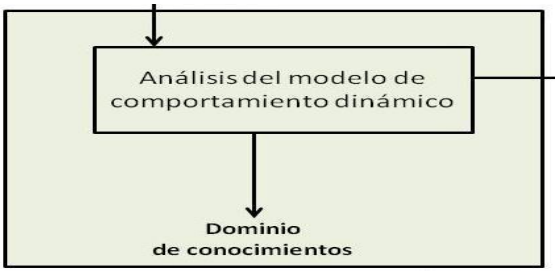

Figura 4. Fase 2 del MECCMA (Roshandel, 2006)

El análisis del modelo de comportamiento dinámico va a estar representado por un grafo dirigido, en el cual cada nodo representa un estado transitorio por el que puede pasar el componente durante un fallo o error. La recuperación y el fallo total son los estados terminales del grafo, y los posibles estados finales del modelo de comportamiento dinámico (Pérez, 2012).

Fase3. Cálculo de la confiabilidad

La tercera fase tiene como salida la confiabilidad del componente (figura 5). El modelo definido tiene como entrada la cuantificación de los errores de la primera fase, y la confiabilidad de cada subcaracterística de la segunda fase. En esta fase se requiere la realización de cálculos, que permitan un análisis de la confiabilidad, de modo en que a partir de este, se puedan identificar los errores más costosos al sistema, y

adoptar decisiones en consecuencia que los mitiguen. La tercera fase del modelo quedaría reestructurada de la siguiente manera (Ver figura 5):

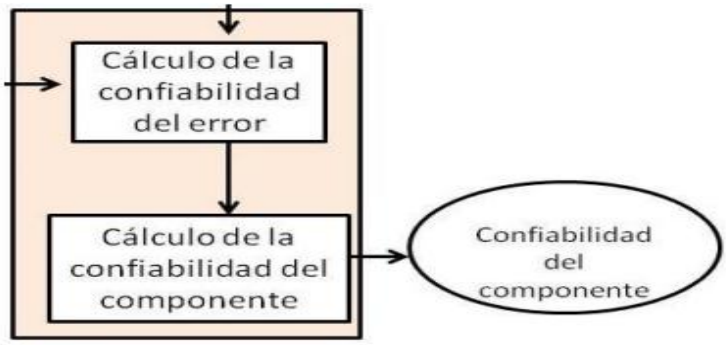

Figura 5. Fase3 del MECCMA (Roshandel, 2006)

En la figura 6 se muestra la estructura del modelo luego de las modificaciones realizadas. En esta figura se divisan las tres fases del modelo y los módulos a realizar en cada una de ellas, hasta llegar a la fase final a partir de la cual se obtiene el valor de la confiabilidad del componente.

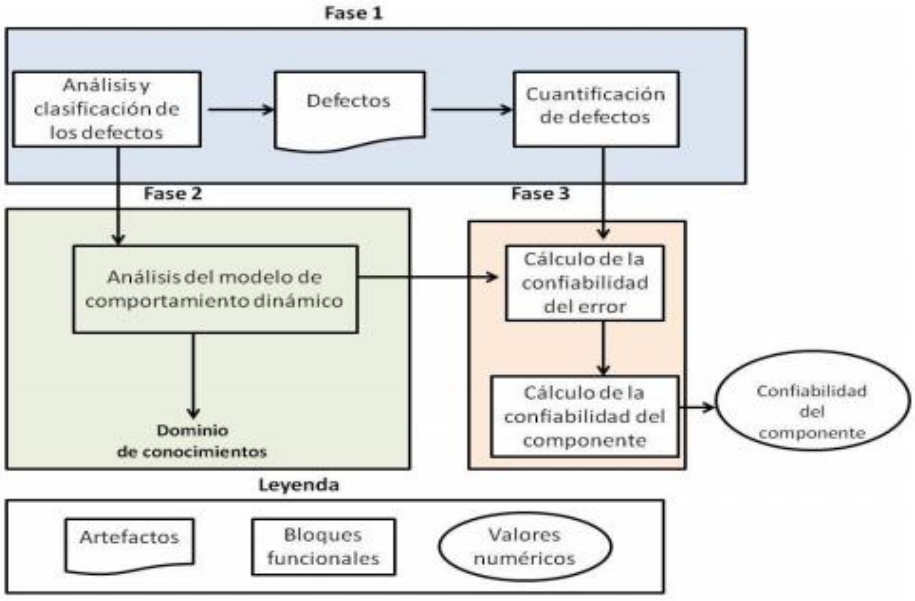

Figura 6. Modelo para evaluar la confiabilidad de un componente de la arquitectura de Cedrux

El modelo definido centra su principal ventaja en brindar resultados que puedan retribuir el proceso de desarrollo. Durante el desarrollo de cada una de las fases por las que se transita para llegar al cálculo final de la confiabilidad, se arriban a conclusiones que son beneficiosas para la toma de decisiones del equipo de desarrollo (Pérez, 2012).

1.3 Herramientas de evaluación cuantitativa

Las herramientas de evaluación cuantitativas son instrumentos de análisis de datos, las cuales pueden anticipar los cambios de manera que se puedan planificar e implementar estrategias que mejoren los resultados. Al aplicar soluciones de análisis a los datos que ya tiene, su organización podrá descubrir patrones y desarrollar productos de forma más eficaz o identificar y minimizar fallos o errores (IBM, 2013). Estas herramientas de evaluación cuantitativas implementan de una forma u otra, diferentes procesos de evaluación como el análisis de evaluación de indicadores de calidad, máquinas de estados, modelos matemáticos, simulaciones y algoritmos.

Weka

Weka, acrónimo de Waikato Environment for Knowledge Analysis, es un entorno para experimentación de análisis de datos que permite aplicar, analizar y evaluar las técnicas más relevantes de análisis de datos, principalmente las provenientes del aprendizaje automático, sobre cualquier conjunto de datos del usuario. Weka se distribuye como software de libre distribución desarrollado en Java, la versión más reciente Weka 3, que empezó a desarrollarse en 1997, se utiliza en muchas y muy diferentes áreas, en particular con finalidades docentes y de investigación (María García Jiménez, 2007) (Ecured, 2013). Está constituido por una serie de paquetes de código abierto con diferentes técnicas de pre-procesado, clasificación, agrupamiento, asociación, y visualización, así como facilidades para su aplicación y análisis de prestaciones cuando son aplicadas a los datos de entrada seleccionados. Dentro de las técnicas de agrupamiento implementa el algoritmo de las K-medias por ser uno de los más veloces y eficientes. Este algoritmo precisa únicamente del número de categorías similares en las que queremos dividir el conjunto de datos. Suele ser de interés repetir la ejecución del algoritmo K-medias con diferentes semillas de inicialización, dada la notable dependencia del arranque cuando no está clara la solución que mejor divide el conjunto de instancias.

AMECAS

El Ambiente de Evaluación de Calidad de Arquitecturas de Software (AMECAS) se define para el contexto de trabajo del Laboratorio de Investigación de Sistemas de Información de la Universidad Simón Bolívar (LISI), con la intención de permitir al Área de Proyectos de Investigación y el Área de Trabajos de Investigación, la construcción de sistemas de calidad, a través de la evaluación de las arquitecturas correspondientes a los proyectos que allí se desarrollan. AMECAS se concibe como un medio que facilite la ejecución de las actividades asociadas a la aplicación de los métodos y técnicas de evaluación de arquitecturas disponibles. De esta manera, es posible aumentar las probabilidades de obtener un diseño idóneo de la arquitectura, además de ofrecer la posibilidad de evaluar una misma arquitectura a través de distintos métodos y técnicas, así como también generar la documentación del proceso de evaluación (Pérez, 2005). AMECAS implementa y soporta los métodos más populares en cuanto a evaluación de arquitecturas, brindando al mismo tiempo soporte a la especificación de la arquitectura y de la calidad. De esta manera, se convierte en una herramienta de apoyo fundamental dentro del proceso de desarrollo, especialmente en las etapas más tempranas.

Valoración de los sistemas estudiados

Después de realizado un estudio a los sistemas informáticos antes mencionados y teniendo en cuenta los aspectos desde el punto de vista del software, del producto, análisis de evaluación de indicadores de calidad (AEIC), tecnología en la que se desarrollan y también de la utilización de máquinas de estado, dentro de la evaluación cuantitativa en dichos sistemas, se pudo establecer un análisis comparativo de los principales parámetros que deben poseer para cumplir con las necesidades y los requisitos necesarios de CEDRUX. Ver Tabla1

Sistemas	Parámetros					
	Software Gratuito	Multiplataforma	AEIC	Máquinas de estado	Técnicas de evaluación cuantitativa	Tecnología en la que esta desarrollada
Weka	si	si	no	no	si	Java
AMECAS	si	si	si	no	No, se basa	Java

					en escenarios	

Tabla 1. Comparación de los sistemas estudiados

Como se evidencia en la tabla 1 ambos sistemas son software gratuito y pueden ser utilizados en cualquier plataforma, pero cuentan con una serie de restricciones:

- El sistema Weka no realiza ningún análisis a los indicadores de calidad, por tanto descarta el objetivo principal de este trabajo.
- El sistema AMECAS emplea indicadores para la evaluación de la calidad, pero la técnica de evaluación es basada en escenarios (cualitativa).

Por ende, ninguno de los dos sistemas compensan en su totalidad las necesidades a satisfacer con la presente investigación; sin embargo un análisis cualitativo de cada uno de ellos permite identificar fortalezas y reutilización para el futuro desarrollo, tales como:

- Weka implementa las K-medias para la minimización de las distancias internas, lo cual se tendrá en cuenta a la hora de desarrollar la aplicación, reutilizando código de dicho algoritmo.
- Ambas aplicaciones están sustentadas sobre el lenguaje java.
- AMECAS se basa en la norma ISO 9126 para la definición de la calidad y aunque hoy sustenta la evaluación por escenarios, propone el uso de modelos matemáticos para evaluar modelos arquitectónicos.

1.4 Metodología de desarrollo de software

La metodología Extreme programming (XP) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, pertenece al grupo de metodologías ágiles muy utilizada para producir software en breves espacios de tiempo y con un equipo reducido y que se enfrenta a requisitos cambiantes. Hace énfasis en la adaptabilidad del proceso. En el desarrollo vincula estrechamente al usuario final con el equipo propiciando un buen clima de trabajo como clave para lograr el éxito. Está basado en 5 valores básicos (José, y otros, 2003):

1. **Simplicidad:** enfocado más en un diseño sencillo del código generando sólo la documentación indispensable.
2. **Comunicación:** potenciada por el desarrollo en pares y la presencia del cliente, además de la simplicidad en cuanto al código.
3. **Retroalimentación:** propiciada por el protagonismo del cliente que participa activamente y por el trabajo en ciclos cortos.
4. **Coraje:** enfrentando decisiones en ocasiones complejas que pudieran afectar el tiempo de desarrollo y la calidad del producto.
5. **Respeto:** basado en estimar en toda su magnitud el trabajo de los demás.

Sustentados en estos 5 valores los proyectos XP deben tener muy presentes 24 prácticas valiosas, que se complementan unas con otras y que ofrecen una base sólida para un óptimo desempeño, alta productividad e inestimables beneficios (Beck, 1999). Entre ellas destacamos:

- Pruebas antes de programar
- Diseño incremental
- Ciclo semanal
- Integración continúa
- Programación en parejas
- Implicación real del cliente

La misma define cuatro fases fundamentales: Exploración, Planificación de la Entrega (Release), Implementación y Prueba.

Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas,

tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología. (Penadés, 2004)

Planificación de la Entrega (Release)

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. (Penadés, 2004)

Implementación

La fase de implementación requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. (Penadés, 2004)

Prueba

En esta fase es donde se le aplican las pruebas de aceptación al sistema por parte del cliente, se busca que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema, además la producción de código está dirigida por las pruebas unitarias. Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. (Penadés, 2004)

- ✓ **Tipos de Pruebas según la metodología XP**

- **Pruebas unitarias:** Las pruebas unitarias son una de las piedras angulares de XP, las mismas consisten en comprobaciones (manuales o automatizadas) desarrolladas por los programadores que se realizan para verificar que el código correspondiente a un módulo concreto se comporta de manera esperada. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, las pruebas deben ser definidas antes de realizar el código. Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la

propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o re-codificar parte del mismo (Joskowicz, 2008). Estas pruebas no generan artefactos y no son directamente palpables para el cliente lo cual no quiere decir que no sean de vital importancia para el desarrollo del proyecto.

- **Pruebas de aceptación:** Las pruebas de aceptación son pruebas que se crean a partir de las historias de usuario. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una HU ha sido implementada correctamente. Una HU puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de estas pruebas es garantizar que los requerimientos hayan sido cumplidos y que la aplicación es realmente lo que el cliente quería. Una HU no se considera terminada hasta que no ha pasado sus pruebas de aceptación. Estas pruebas son más importantes que las pruebas unitarias, dado que significan la satisfacción del cliente con el producto desarrollado al final de una iteración.

1.5 Herramientas y tecnologías para el desarrollo

Lenguaje Unificado de Modelado (UML 6.4)

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema, documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar (Peter, 2003).

Java

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos

más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria (ANDINO, 2011). Java ofrece una potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable, elimina ciertas complejidades como pueden ser el manejo de memoria, no depende de una arquitectura computacional ya que el compilador de Java genera un código conocido como byte code que puede ser interpretado por cualquier computadora una vez que se haya implementado un intérprete par cada plataforma, es gratis, universal y de fácil distribución.

PL/pgSQL

PL/pgSQL Lenguaje de procedimientos almacenados para Postgres (Procedural Language/Postgres Structured Query Language en inglés) es un lenguaje provisto por el gestor de base de datos PostgreSQL donde se pueden realizar cálculos complejos y crear nuevos tipos de datos de usuario, dispone de estructuras de control repetitivas y condicionales, además de la posibilidad de creación de funciones que pueden ser llamadas en sentencias SQL normales o ejecutadas en eventos de tipo disparador (trigger) y que herede todos los tipos definidos por el usuario, las funciones y los operadores (Martinez, 2013). Una de las principales ventajas de ejecutar programación en el servidor de base de datos es que las consultas y el resultado no tienen que ser transportadas entre el cliente y el servidor, ya que los datos residen en el propio servidor. Además, el gestor de base de datos puede planificar optimizaciones en la ejecución de la búsqueda y actualización de datos.

Herramientas

Herramientas **CASE** (Computer Aided Software Engineering por sus siglas en inglés): conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar el proceso de diseño y desarrollo de software.

Visual Paradigm for UML 8.0

Visual Paradigm es una de las herramientas UML CASE del mercado, considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones ejemplo NetBeans IDE. Fue creada para el ciclo vital completo del desarrollo de

software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación. Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de clases, es compatible entre ediciones. Presenta una licencia gratuita y comercial.

NetBeans IDE 7.1.2

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso (NetBeans, 2013). Es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (Object Relational Database Management System (ORDBMS) por sus siglas en inglés) basado en el proyecto POSTGRES, de la universidad de Berkeley, es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje PL/pgSQL. Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional (PostgreSQL-es.org).

1.6 Conclusiones parciales

A lo largo de este capítulo se han ofrecido los elementos teóricos que sirven de sustento a la investigación, para así darle solución a los objetivos planteados, arribando a las siguientes conclusiones:

- Se realizó un estudio por cada una de las fases de dicho modelo incluyendo las modificaciones realizadas en el mismo por (Pérez, 2012) arrojando como resultado la comprensión de dicho modelo.
- Se realizó un estudio de algunas herramientas de evaluación cuantitativas y se concluyó que ninguna cumple con el objetivo de este trabajo.
- Se realizó un estudio de la metodología XP y se revisaron todas sus potencialidades y se seleccionó la misma para guiar el proceso de desarrollo de la aplicación.

CAPÍTULO 2. DISEÑO E IMPLEMENTACION DEL SISTEMA

El presente capítulo se enfocará en los detalles técnicos de la solución que se desea implementar. Se describen los principales artefactos que define la metodología seleccionada para llevar a cabo el desarrollo de la aplicación, así como la descripción de estos. Se centra las características generales del sistema, la captura y especificación de los requisitos funcionales.

2.1 Propuesta del sistema

La solución que se propone está diseñada para la evaluación cuantitativa de la confiabilidad de la arquitectura de Cedrux, siendo una aplicación de apariencia profesional y un diseño gráfico sencillo, posibilitando al usuario que inserte, modifique o elimine errores de la arquitectura del proyecto, además permite el análisis del modelo de comportamiento dinámico a través de un grafo dirigido y también realiza el cálculo de la confiabilidad de los componentes del proyecto. El sistema cuenta con una base de datos en la cual queda almacenada toda la información introducida por el usuario y este último será capaz de verificar la confiabilidad de la arquitectura de los componentes evaluados, arrojando así datos precisos con los cuales se podrá tomar una decisión a la hora de la evaluación de la arquitectura de Cedrux.

2.2 Fase de exploración

Requisitos Funcionales

Los requisitos funcionales representan el comportamiento que tendrá el sistema. Describen las funcionalidades que debe cumplir o que se espera que este provea. Deben ser lo más completos, claros y concisos posibles.

RF.01: Insertar Proyecto.

RF.02: Modificar Proyecto.

RF.03: Eliminar Proyecto.

RF.04: Insertar Componente Primario.

RF.05: Modificar Componente Primario.

RF.06: Eliminar Componente Primario.

RF.07: Insertar Componente Secundario.

RF.08: Modificar Componente Secundario.

RF.09: Eliminar Componente Secundario.

RF.10: Insertar Error.

RF.11: Modificar Error.

RF.12: Eliminar Error.

RF.13: Obtener coeficiente de confiabilidad de Madurez.

RF.14: Obtener coeficiente de confiabilidad de Tolerancia a fallos.

RF.15: Obtener coeficiente de confiabilidad de la Recuperabilidad.

RF.16: Obtener coeficiente de confiabilidad del componente ante el error o fallo.

RF.17: Clasificar la confiabilidad del error.

RF.18: Obtener coeficiente de confiabilidad del componente.

RF.19: Clasificar la confiabilidad del componente.

RF.20: Generar confiabilidad de los componentes evaluados.

Descripción de los requisitos no funcionales de software

Requisitos No Funcionales: Los requisitos no funcionales establecen las características o propiedades que debe tener el sistema. Muchos requerimientos no funcionales se refieren al sistema como un todo

más que a rasgos particulares del mismo. Para la definición de los requisitos no funcionales se utilizará la clasificación de la Norma ISO-9126.

- **Usabilidad**

RnF.01: La herramienta debe visualizar todos los mensajes en idioma español. La tipografía debe ser uniforme (Times New Roman, tamaño 9).

- **Eficiencia**

RnF.02: La mayoría de los procesos que se implementan con transacciones donde se modifica la base de datos deben tener tiempos de respuesta no mayores a los 3 segundos. En el caso de la obtención de información que involucran consultas a la base de datos, se busca que el tiempo de respuesta se simplifique al máximo y no exceda los 5 segundos.

- **Portabilidad**.

RnF.03: La herramienta se desarrollará con tecnología Java.

RnF.04: La herramienta utilizará una base datos implementada en PostgreSql versión 8.3.

RnF.05: La herramienta estará disponible para versiones de Windows 7 o inferior, así como Linux, todas con sus versiones actuales.

- **Precisión**

RnF.06: Los resultados ofrecidos por el sistema respecto a valores calculados tendrán una exactitud con un margen de error no mayor que 0,10.

RnF.07: Los resultados ofrecidos por el sistema respecto a valores calculados serán redondeados a dos cifras después de la coma.

CAPÍTULO 2: **DISEÑO E IMPLEMENTACIÓN DEL SISTEMA**

Descripción de las historias de los usuarios

Entre los artefactos generados por la metodología XP se pueden encontrar las historias de usuarios (HU). Las HU son creadas a partir de las funcionalidades definidas. Las siguientes tablas describen las principales HU de los diferentes Requisitos Funcionales (RF), que presentan prioridad alta conformando parte de las funcionalidades del sistema. Las HU (tabla 2, tabla 3, tabla 4) se corresponden con los requisitos “10”, “16”, “18”. Las restantes HU pueden encontrarse en el anexo 1.

Se realiza la descripción de las historias de los usuarios y su responsable asignado para el cumplimiento de la misma.

Historia de Usuario	
Código: HU10	**Nombre Historia de Usuario:** Insertar Errores
Modificación de Historia de Usuario Número: 1	
Referencia: RF.10	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite al usuario la opción de insertar errores, requiriéndose como datos de entra (el nombre del proyecto, el nombre del componente primario o secundario y la descripción del error). El sistema debe informar que se realizó satisfactoriamente la operación escogida.	
Observaciones: En caso de dejar algunos de los campos en blanco se muestra un mensaje “Debe llenar todos los campos”.	

Tabla 2: Descripción de la HU Insertar Errores

Historia de Usuario	
Código: HU16	**Nombre Historia de Usuario:** Obtener coeficiente de confiabilidad del componente ante el error o fallo.
Modificación de Historia de Usuario Número: 1	
Referencia: RF.16	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite mostrar al usuario la confiabilidad del componente ante el error o fallo dependiendo de los cálculos realizados permitiendo un análisis de la confiabilidad, teniendo como entrada la cuantificación de errores de la primera fase del modelo y la confiabilidad de cada subcaracterística de la segunda fase.	
Observaciones:	

Tabla 3: Descripción de la HU Obtener coeficiente de confiabilidad del componente ante el error o fallo

Historia de Usuario	
Código: HU18	**Nombre Historia de Usuario:** Obtener coeficiente de confiabilidad del componente.
Modificación de Historia de Usuario Número: 1	
Referencia: RF.18	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana

Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta realiza una serie de cálculos utilizando los datos anteriormente entrados en las distintas fases del modelo obteniendo la confiabilidad del componente.	
Observaciones:	

Tabla 4: Descripción de la HU Obtener coeficiente de confiabilidad del componente

2.3 Fase Planificación

Plan de entrega

En esta fase se establece la prioridad de cada historia de usuario así como se realiza una estimación del esfuerzo necesario de cada una de ellas, se toman acuerdos sobre el contenido de la entrega y se determina un cronograma en conjunto con el cliente. Las estimaciones de esfuerzo asociado a la implementación de las historias se establecen utilizando como medida el punto, este es considerado como una semana de trabajo, semanas de 40 horas, en la que el equipo de desarrollo labora de forma ininterrumpida.

No	Historia de Usuario	Puntos Estimados
1	Insertar Proyecto.	1
2	Modificar Proyecto.	1
3	Eliminar Proyecto.	1
4	Insertar Componente Primario.	1
5	Modificar Componente Primario.	1
6	Eliminar Componente Primario.	1

7	Insertar Componente Secundario.	1
8	Modificar Componente Secundario.	1
9	Eliminar Componente Secundario.	1
10	Insertar Error.	1
11	Modificar Error.	1
12	Eliminar Error.	1
13	Obtener coeficiente de confiabilidad de Madurez.	1
14	Obtener coeficiente de confiabilidad de Tolerancia a fallos.	1
15	Obtener coeficiente de confiabilidad de la Recuperabilidad.	1
16	Obtener coeficiente de confiabilidad del componente ante el error o fallo.	1
17	Clasificar la confiabilidad del error.	1
18	Obtener coeficiente de confiabilidad del componente.	1
19	Clasificar la confiabilidad del componente.	1
20	Generar confiabilidad de los componentes evaluados.	1

Tabla 5. Plan de esfuerzo por Historia de usuario

CAPÍTULO 2: **DISEÑO E IMPLEMENTACIÓN DEL SISTEMA**

Plan de Iteraciones

Luego de identificadas las historias de usuario y estimado el esfuerzo dedicado a la realización de cada una de estas historias se procede a la realización de la planificación de la etapa de implementación del sistema a desarrollar. En base a lo antes mencionado se decide realizar esta en cinco iteraciones, las cuales se detallan a continuación (Penadés, 2004).

Iteración 1: En la primera iteración se realiza la implementación de las HU 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 obteniendo al final de la misma una primera versión de prueba y dando a la aplicación las primeras funcionalidades.

Iteración 2: En la segunda iteración se implementa la HU 13, 14, 15. Además se corregirán errores o disconformidades del cliente con las HU implementadas en la anterior iteración. De esta forma se obtiene la segunda versión de prueba de la aplicación.

Iteración 3: En la tercera iteración se realiza la implementación de las HU 16, 17. Además se corregirán errores o disconformidades del cliente con las HU implementadas anteriormente. De esta forma se obtiene la tercera versión de prueba de la aplicación.

Iteración 4: En la cuarta iteración se realiza la implementación de las HU 18, 19, 20. Además se corregirán errores o disconformidades del cliente con las HU implementadas anteriormente. Al final de esta iteración se obtendrá la versión final de la aplicación.

Plan de duración de las iteraciones

Este plan se realiza con el objetivo de reflejar cuáles serán las HU que serán implementadas en cada una de las iteraciones, así como el tiempo (en semanas) destinado a cada una de ellas y el orden en que se implementarán.

Iteración	Orden de las Historias de Usuarios	Duración de las Iteraciones
Iteración 1	Insertar Proyecto. Modificar Proyecto. Eliminar Proyecto. Insertar Componente Primario. Modificar Componente Primario. Eliminar Componente Primario. Insertar Componente Secundario. Modificar Componente Secundario. Eliminar Componente Secundario. Insertar Error. Modificar Error. Eliminar Error.	3 semanas
Iteración 2	Obtener coeficiente de confiabilidad de Madurez Obtener coeficiente de confiabilidad de Recuperabilidad Obtener coeficiente de confiabilidad de Tolerancia a fallos	3 semanas
Iteración 3	Obtener coeficiente de confiabilidad del componente ante el error	2 semanas

	o fallo Clasificar la confiabilidad del error	
Iteración 4	Obtener coeficiente de confiabilidad del componente Clasificar la confiabilidad del componente Generar confiabilidad de los componentes evaluados.	2 semanas

Tabla 6. Plan de duración de las iteraciones

2.4 Fase Implementación

Durante el transcurso de las iteraciones se realiza la implementación de las historias de usuario seleccionadas para cada una de estas. Al inicio de las mismas, se lleva a cabo una revisión del plan de iteraciones y se modifica de ser necesario.

Tarjetas Cargo o Clase, Responsabilidad y Colaboración (Tarjetas CRC)

El diagrama de tarjetas CRC es una técnica de modelado orientado a objetos que permite identificar las clases y sus responsabilidades. Las tarjetas CRC son una herramienta de ayuda al refinamiento de clases. Consiste en elaborar para cada clase una tarjeta con los siguientes datos: Nombre, Responsabilidades, Colaboraciones.

Las tarjetas determinan el comportamiento de cada actividad. Las tarjetas CRC que se muestra a continuación pertenece a la clase Error, las demás identificadas durante el desarrollo del sistema se encuentran anexas, específicamente en el Anexo 2.

Tarjeta CRC	
Clase: Error	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades:	**Colaboraciones:**
Insertar Error Modificar Error Eliminar Error	ComponenteN1 ComponenteN2 NConfiabilidadErrorComp NtipoError Recorrido

Figura 7. Tarjeta CRC (Error)

Esta clase es la encargada de insertar los errores teniendo en cuenta el proyecto, el componente nivel 1 o nivel 2 al que pertenece, la descripción del error y el tipo de error al que pertenece. Además es la responsable de la modificación de los errores insertados y de la eliminación de dichos errores. Colaborando con las clases NtipoError, Recorrido, ComponenteN1, ComponenteN2 y NConfiabilidadErrorComp.

Representación del diagrama de Base de Datos

El diseño de la base de datos es una tarea de vital importancia para la correcta implementación, en esta se describen las relaciones y los tipos de datos que se van a usar para lograr un correcto funcionamiento del sistema. En el caso de la aplicación que se está desarrollando se identificaron 12 tablas. A continuación se muestra el diagrama Entidad-Relación diseñado para la aplicación:

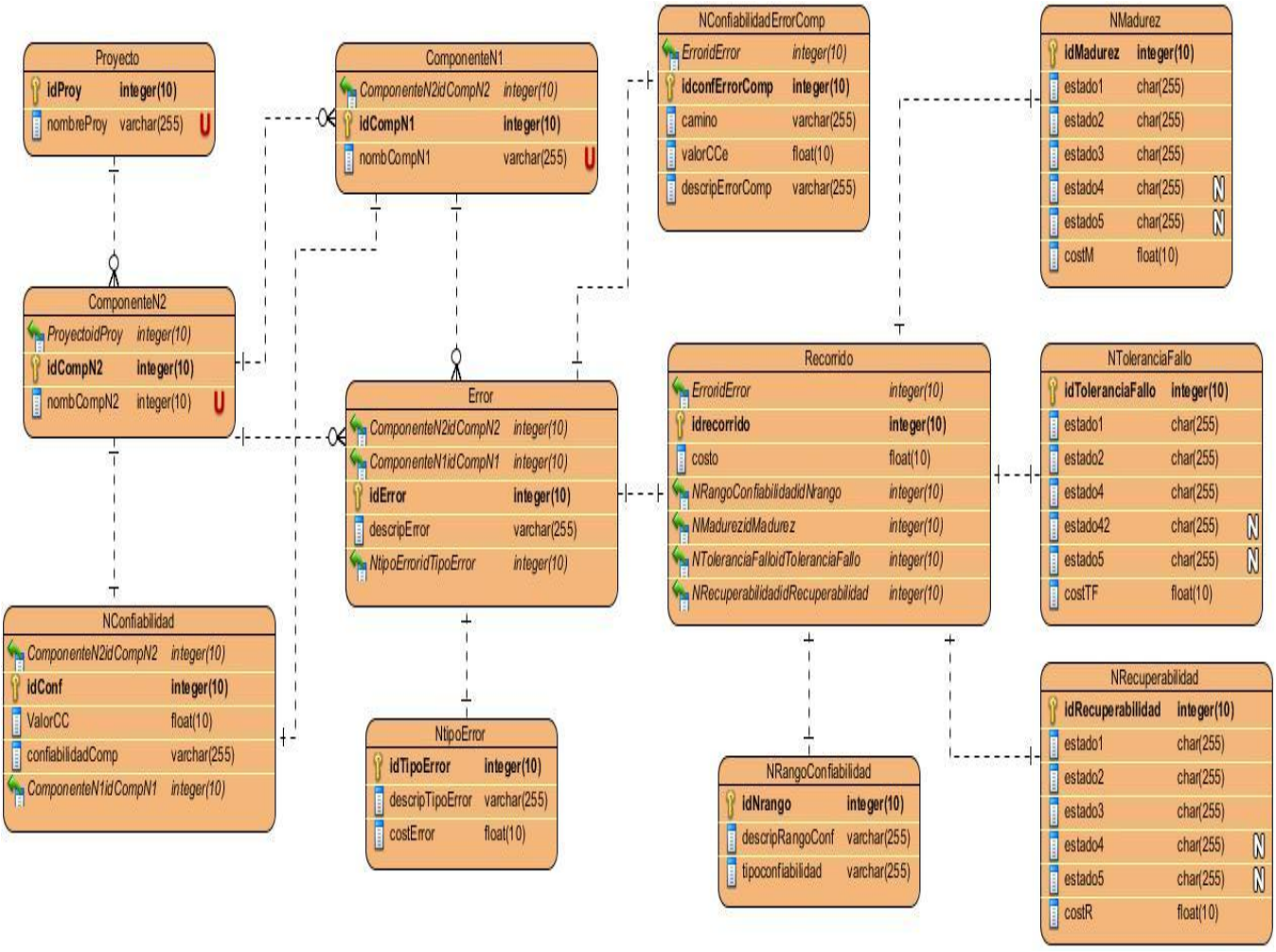

Figura 8. Representación del diagrama de Base de Datos

Descripción de las tablas

Tabla Proyecto: lista todos los proyectos existentes.

Tabla ComponenteN2: lista de componentes principales que tiene el proyecto.

Tabla ComponenteN1: lista de componentes secundarios que tiene el componente primario.

Tabla Error: lista de errores de cada uno de los componentes primarios.

Tabla NtipoError: permite cuantificar los errores, según el tipo de error que se seleccione.

Tabla Recorrido: permite la gestión de cada error en la fase de análisis del modelo de comportamiento dinámico.

Tabla NConfiabilidad: lista la confiabilidad de cada uno de los componentes evaluados.

Tabla NConfiabilidadErrorComp: lista la confiabilidad de cada error ante el componente al que pertenece.

Tabla NRangoConfiabilidad: nomenclador que permite saber la confiabilidad de un error dependiendo del rango en que se encuentre el error.

Tabla NMadurez: nomenclador que permite saber la confiabilidad de la madurez de un error dependiendo el camino que tome el error en el grafo del análisis del comportamiento dinámico.

Tabla NToleranciaFallo: nomenclador que permite saber la confiabilidad de la tolerancia a fallo de un error dependiendo el camino que tome el error en el grafo del análisis del comportamiento dinámico.

Tabla NRecuperabilidad: nomenclador que permite saber la confiabilidad de la recuperabilidad de un error dependiendo el camino que tome el error en el grafo del análisis del comportamiento dinámico.

2.5 Fase Prueba

El objetivo de las pruebas de software es detectar el mayor número posible de errores. Para lograrlo existen diferentes técnicas. Por otro lado, el proceso de pruebas no se debe plantear exclusivamente en las fases finales de la implementación del producto, incluso algunos autores definen la necesidad de dar la vuelta completamente al modelo de desarrollo lineal y comenzar la elaboración del software por la especificación de los casos de prueba (Tuya, 2007).

XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores y encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar

si al final de una iteración se consiguió la funcionalidad requerida además de comprobar que dicha funcionalidad sea la esperada por el cliente (José Antonio, 2007).

- ✓ **Resultados de las Pruebas realizadas**

Cantidad de Pruebas Realizadas.	**Cantidad de Pruebas Satisfactorias.**	**Cantidad de Errores Encontrados.**	**Cantidad de Errores Corregidos.**
15	11	4	4

Tabla 7. Cantidad de Pruebas Unitarias realizadas

Casos de Prueba	
Número de caso de prueba :CP_1	**Número de Historia de Usuario:HU1**
Nombre del Caso de prueba: Adicionar error.	
Condiciones de ejecución: El usuario debe insertar un error a un componente secundario teniendo en cuenta el proyecto y el componente primario al que pertenece el componente secundario.	
Escenario de prueba 1	
Entradas: Introducción correcta de los datos.	
Resultado esperado: El error es agregado correctamente.	
Escenario de prueba 2	
Entradas: Dejar datos sin seleccionar.	
Resultado esperado: Se muestra un mensaje indicando que “Debe seleccionar todos los campos y llenar la descripción del error”, permitiendo seleccionar nuevamente los datos.	
Escenario de prueba 3	
Entradas: No seleccionar datos.	
Resultado esperado: Se muestra un mensaje indicando que “Debe seleccionar todos los campos y	

llenar la descripción del error", permitiendo seleccionar nuevamente los datos.
Evaluación: Satisfactoria.

Tabla 8: Caso de prueba de aceptación HU1

Casos de Prueba	
Número de caso de prueba :CP_2	**Número de Historia de Usuario:HU1**
Nombre del Caso de prueba: Modificar error.	
Condiciones de ejecución: El usuario debe seleccionar el error que desea modificar y dar click en el botón aceptar, saliendo una nueva ventana la cual le da la opción de modificar el Tipo de Error y la descripción del error.	
Escenario de prueba 1	
Entradas: Seleccionar el error y dar click en el botón aceptar.	
Resultado esperado: Se levanta una nueva venta que permite modificar el error.	
Escenario de prueba 2	
Entradas: No seleccionar el error y dar click en el botón aceptar.	
Resultado esperado: Se muestra un mensaje indicando que "Debe seleccionar al menos un error", permitiendo seleccionar nuevamente otro error.	
Escenario de prueba 3	
Entradas: No modificar ningún campo.	
Resultado esperado: Se muestra un mensaje indicando que "No se ha modificado ningún campo", permitiendo modificar los campos.	
Escenario de prueba 4	
Entradas: Modificar al menos un campo.	
Resultado esperado: Se muestra un mensaje indicando que "Se ha modificado correctamente", permitiendo seleccionar un nuevo error para ser modificado.	

Evaluación: Satisfactoria.

Tabla 9: Caso de prueba de aceptación HU1

Casos de Prueba	
Número de caso de prueba :CP_3	**Número de Historia de Usuario:HU1**
Nombre del Caso de prueba: Eliminar error.	
Condiciones de ejecución: El usuario debe seleccionar el error que desea eliminar y dar click en el botón aceptar, eliminando el error de la aplicación.	
Escenario de prueba 1	
Entradas: Seleccionar el error y dar click en el botón aceptar.	
Resultado esperado: Se elimina el error de la aplicación permitiéndole eliminar otros errores.	
Escenario de prueba 2	
Entradas: No seleccionar el error y dar click en el botón aceptar.	
Resultado esperado: Se muestra un mensaje indicando que “Debe seleccionar al menos un error”, permitiendo seleccionar nuevamente otro error.	
Evaluación: Satisfactoria.	

Tabla 10: Caso de prueba de aceptación HU1

2.6 Patrones de diseño utilizados

El diseño propuesto fue creado siguiendo patrones que de manera general constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, lo cual permitirá llevar una mejor implementación de la aplicación. En este caso se emplearon algunos de los patrones generales de software para asignación de responsabilidades (**GRASP** acrónimo en inglés), los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones, según lo indicado por (Larman, 1999). Los patrones GRASP que se utilizados son los siguientes:

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa (Larman, 1999), de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado, el cual se evidencia en la clase Reporte1.

Creador: Plantea la asignación de responsabilidad a una clase B para crear una instancia de clase A (Larman, 1999). Este patrón es adaptable a la clase Conectar, la cual es la encargada de crear los objetos de tipo PreparedStatement, createStatement, conexion, para permitir el acceso a la información almacenada a nivel de datos.

2.7 Conclusiones parciales

A lo largo de este capítulo se realizó todo lo referente a las etapas de exploración, planificación, implementación y pruebas que propone la metodología seleccionada:

- Se realizaron pruebas unitarias y de aceptación arrojando como resultado un código más depurado proporcionando más facilidad para darle soporte a la herramienta y se le presentó la misma al cliente observándose el grado de conformidad a este último.
- Con la solución propuesta se podrá medir la confiabilidad de los componentes del sistema Cedrux y será de ayuda para tomar decisiones arquitectónicas a los arquitectos del mismo, a través del cálculo de la confiabilidad del error ante el componente y la confiabilidad del componente en general según los cálculos antes mencionados.

CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

El presente capítulo tiene como principal objetivo la validación de la solución propuesta. Para ello se verifica la precisión de los datos obtenidos mediante la comparación entre datos esperados y datos obtenidos. Se realiza un análisis de los componentes escogidos para la muestra y finalmente se analizan los datos resultantes de la ejecución de la herramienta.

3.1 Caracterización del entorno de prueba

La población está constituida por un total de 12 componentes de nivel 2 de Cedrux, pero solo se le aplica a 9 de ellos que están siendo utilizados actualmente. En la tabla 11, se exponen los elementos de la muestra (Pérez, 2012):

No.	Componentes
1	Inventario
2	Facturación
3	Activos Fijos
4	Finanzas
5	Auditoria
6	Configuración
7	Contabilidad
8	Capital Humano
9	Marco de Trabajo

Tabla 11. Componentes de la muestra

3.2 Análisis de indicadores

Para el análisis de la variable precisión se utiliza la métrica establecida por la norma ISO 9126. La misma plantea que la exactitud viene dada por la: *"capacidad del software para proporcionar efectos o resultados correctos o convenidos con el grado de exactitud necesario"* (ISO, 2005). Teniendo en cuenta los requisitos no funcionales del sistema, se toma para esta investigación como valor exacto todo aquel que cumpla que: la diferencia entre los valores esperados (X) y los valores actuales (Y) sean menores o igual a 0.1 (|X-Y| <= 0.10). La métrica se aplicaría para cada uno de los valores calculados del sistema acorde a la muestra. Esto queda reflejado en la tabla 12.

Nombre de la métrica	**La métrica se propone medir**	**Medición (fórmula)**	**Interpretación del valor obtenido**	**Escala**
Exactitud esperada	**¿Existen diferencias entre los valores actuales y los razonablemente esperados?**	**A = B/C** **B - Número de casos encontrados sin diferencias entre los valores esperados y los actuales.** **C - Número de casos probados.**	**0 <= A<=1** **A mayor cercanía al 1 mayor precisión de los datos obtenidos.**	**Valorativa**

Tabla 12. Métrica de precisión

3.3 Datos del entorno de prueba

La problemática descrita en la presente investigación incide en la precisión de los datos obtenidos, esto viene dado por los procesamientos no triviales y en algunos casos voluminosos a realizar con la utilización de MECCMA. Estos cálculos se llevan a cabo en las fases 2 y 3 del Modelo, para la determinación de la confiabilidad del error ante el componente y la confiabilidad total del componente, respectivamente.

Teniendo estos elementos en cuenta y la descripción de las variables a analizar en la métrica presentada en el epígrafe 3.2, se decide tomar como valores esperados (X) los previamente calculados en (Pérez, 2012), pero solo los refereridos a los valores calculados (confiabilidad del error y confiabilidad del componente). Estos valores son los subrayados en las tablas 13 a la 20.

Subsistema: Inventario - Activos Fijos							
No	**Descripción**	**Peso del Error**	**Camino que recorre**	**Cálculo de las subcaracterísticas**			**Confiabilidad**
				M	**TF**	**R**	
1.	Creación de instancias de objetos globales (integrator, común, etc.) en clases del negocio.	0.1	E-TC-R	1	1	1	0.10
2.	Violación de las capas arquitectónicas. Ejemplo: invocación desde una clase controladora a una clase de acceso a datos; clase controladora ejecutando funciones del negocio.	0.1	E-TC-R	1	1	1	0.10
3.	Incorrecto uso de la Herencia. Existen clases hijas que reproducen funciones de la clase padre, clases padre que no implementan funciones que son utilizadas por sus hijas.	0.2	E-TC-R	1	1	1	0.20

4.	Bajo nivel de reutilización de código para la ejecución de acciones con alto nivel de presencia en el negocio.	0.2	E-PS-R	2.5	3	2.5	0.53
Centroides y confiabilidad del subsistema:				**C1**	**C2**	**C3**	**0.83**
				0.10	0.20	0.53	

Tabla 13. Cálculo de la confiabilidad para los subsistemas Inventario – Activos Fijos (Pérez, 2012)

Subsistema: Auditoría							
No	**Descripción**	**Peso del Error**	**Camino que recorre**	**Cálculo de las subcaracterísticas**			**Confiabilidad**
				M	**TF**	**R**	
1	En el componente analizador existen diferentes clases que contienen los mismos métodos debido al mecanismo de integración.	0.1	E_TC _R	1	1	1	0.10
2.	En la interfaz general existe pérdida de etiquetas, debido a la utilización de alertas, y no el mecanismo que define la arquitectura para ser usado.	0.2	E_DA_R	3	2	2	0.47
3.	En la clase controladora Exportar existe implementada lógica del negocio.	0.1	E_TC_R	1	1	1	0.10
Centroides y confiabilidad del subsistema:				**C1**	**C2**	**C3**	**0.57**
				0.10	0.47	-	

Tabla 14. Cálculo de la confiabilidad para los subsistemas Auditoría (Pérez, 2012)

Subsistema: Configuración							
No	**Descripción**	**Peso del Error**	**Camino que recorre**	**Cálculo de las subcaracterísticas**			**Confiabilidad**
				M	**TF**	**R**	
1.	Cuando una entidad es hija (económicamente) de otra, debe heredar automáticamente de esta, los formatos, actualmente cuando ocurre esta acción el sistema duplica uno de los formatos heredados.	0.8	E_DA_R	3	2	2	1.87
2.	Cuando un documento primario está siendo utilizado por algún subsistema, el componente está permitiendo que el mismo sea modificado, causando inconsistencia en la información.	0.8	E_DNA_R	5	7	5	4.53
3.	Implementaciones de lógica en las clases controladoras.	0.1	E_TC_R	1	1	1	0.10
Centroides y confiabilidad del subsistema:				**C1**	**C2**	**C3**	**6.5**
				0.10	1.87	4.53	

Tabla 15. Cálculo de la confiabilidad para los subsistemas Configuración (Pérez, 2012)

Subsistema: Contabilidad							
No	**Descripción**	**Peso del Error**	**Camino que recorre**	**Cálculo de las subcaracterísticas**			**Confiabilidad**
				M	**TF**	**R**	

1.	Se hace uso del fetchAll en vez del query buscando mejorar el rendimiento de la aplicación por la gran cantidad de datos a cargar de otros componentes del sistema.	0.2	E_TC_R	1	1	1	0.20
Centroides y confiabilidad del subsistema:				**C1**	**C2**	**C3**	
				0.20	-	-	**0.20**

Tabla 16. Cálculo de la confiabilidad para los subsistemas Contabilidad (Pérez, 2012)

Subsistema: Facturación							
No	**Descripción**	**Peso del Error**	**Camino que recorre**	**Cálculo de las subcaracterísticas**			**Confiabilidad**
				M	**TF**	**R**	
1.	Violación de las capas arquitectónicas. Ejemplo: invocación desde una clase controladora a una clase de acceso a datos; clase controladora ejecutando funciones del negocio.	0.1	E-TC-R	1	1	1	0.10
2.	Incorrecto uso de la Herencia. Existen clases hijas que reproducen funciones de la clase padre, clases padre que no implementan funciones que son utilizadas por sus hijas.	0.2	E-TC-R	1	1	1	0.20
3.	Bajo nivel de reutilización de código para la ejecución de acciones con alto nivel de presencia en el negocio.	0.2	E-PS-R	2.5	3	2.5	0.53
Centroides y confiabilidad del subsistema:				**C1**	**C2**	**C3**	
				0.10	0.20	0.53	**0.83**

Tabla 17. Cálculo de la confiabilidad para los subsistemas Facturación (Pérez, 2012)

Subsistema: Capital Humano							
No	**Descripción**	**Peso del Error**	**Camino que recorre**	**Cálculo de las subcaracterísticas**			**Confiabilidad**
				M	**TF**	**R**	
1.	Pérdida de etiquetas: al cargar una nueva interfaz y cargar su correspondiente json con sus etiquetas, si se regresa a una interfaz abierta anteriormente no se encuentran las correspondientes a esta.	0.2	E_DA_R	3	2	2	0.47
2.	Clases controladoras conteniendo funcionalidades del negocio.	0.1	E_TC_R	1	1	1	0.10
Centroides y confiabilidad del subsistema:				**C1**	**C2**	**C3**	
				0.10	0.47	-	**0.57**

Tabla 18. Cálculo de la confiabilidad para los subsistemas Capital Humano (Pérez, 2012)

Subsistema: Finanzas							
No	**Descripción**	**Peso del Error**	**Camino que recorre**	**Cálculo de las subcaracterísticas**			**Confiabilidad**
				M	**TF**	**R**	
1.	Llamadas directamente a métodos en las clases Domain desde las Controllers o Services sin pasar por las Models.	0.1	E-TC-R	1	1	1	0.10
2.	No se realizan las validaciones mediante el empleo de weaver.xml y validation.xml.	0.1	E-TC-R	1	1	1	0.10

3.	No se tratan los textos de las etiquetas de la interfaz a través del json y en ocasiones cuando se hace no muestra el texto.	0.2	E-DA-R	3	2	2	0.47
4.	Clases Controllers y Services conteniendo funcionalidades de negocio.	0.2	E-DA-R	3	2	2	0.47
Centroides y confiabilidad del subsistema:				**C1**	**C2**	**C3**	
				0.10	0.47	-	**0.57**

Tabla 19. Cálculo de la confiabilidad para los subsistemas Finanzas (Pérez, 2012)

Subsistema: Marco de Trabajo							
No	**Descripción**	**Peso del Error**	**Camino que recorre**	**Cálculo de las subcaracterísticas**			**Confiabilidad**
				M	**TF**	**R**	
1.	Al existir diferentes conexiones a base de datos para cada subsistema no se puede garantizar la transacionalidad de los procesos que involucran a más de un subsistema.	0.8	E_DNA_PI_R	7	8	6.5	5.73
2.	Las configuraciones de los componentes no están contenidas en los mismos sino en ficheros globales, por lo que un error de configuración de un componente provoca que la aplicación deje de funcionar.	0.2	E_PS_R	2.5	3	2.5	0.53
3.	Existen clases de los componentes que manejan las conexiones a la base de datos, cuando eso es una responsabilidad del marco de trabajo.	0.2	E_TC_R	1	1	1	0.20

4.	Las responsabilidades de las clases están mezcladas existiendo clases de negocio e incluso controladoras implementando consultas o controladoras implementando negocio	0.2	E_TC_R	1	1	1	0.20
				C1	**C2**	**C3**	
Centroides y confiabilidad del subsistema:				0.20	0.53	5.73	**6.46**

Tabla 20. Cálculo de la confiabilidad para los subsistemas Marco de Trabajo (Pérez, 2012)

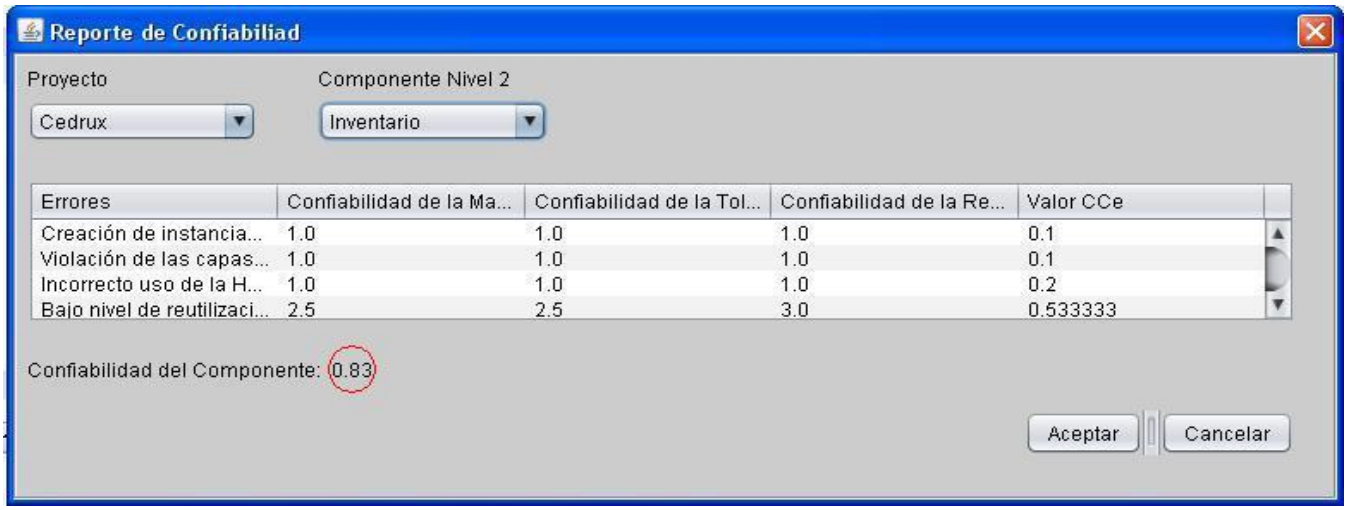

Figura 9. Cálculo de la confiabilidad para los subsistemas Inventario – Activos Fijos

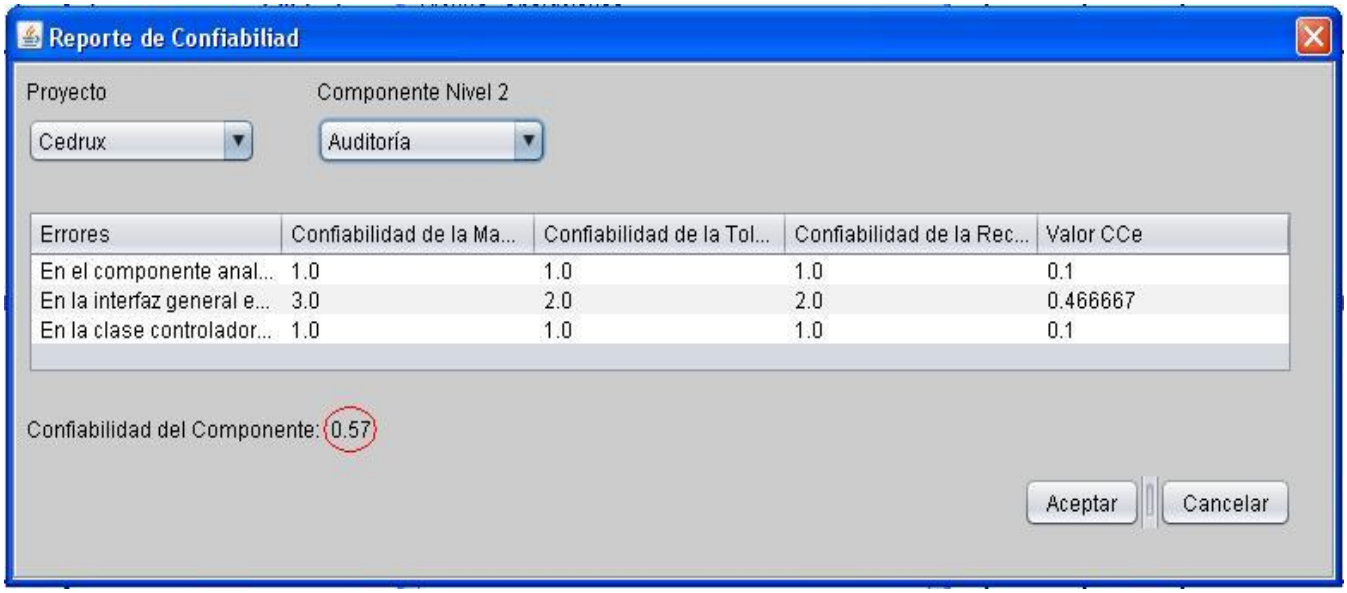

Figura 10. Cálculo de la confiabilidad para los subsistemas Auditoría

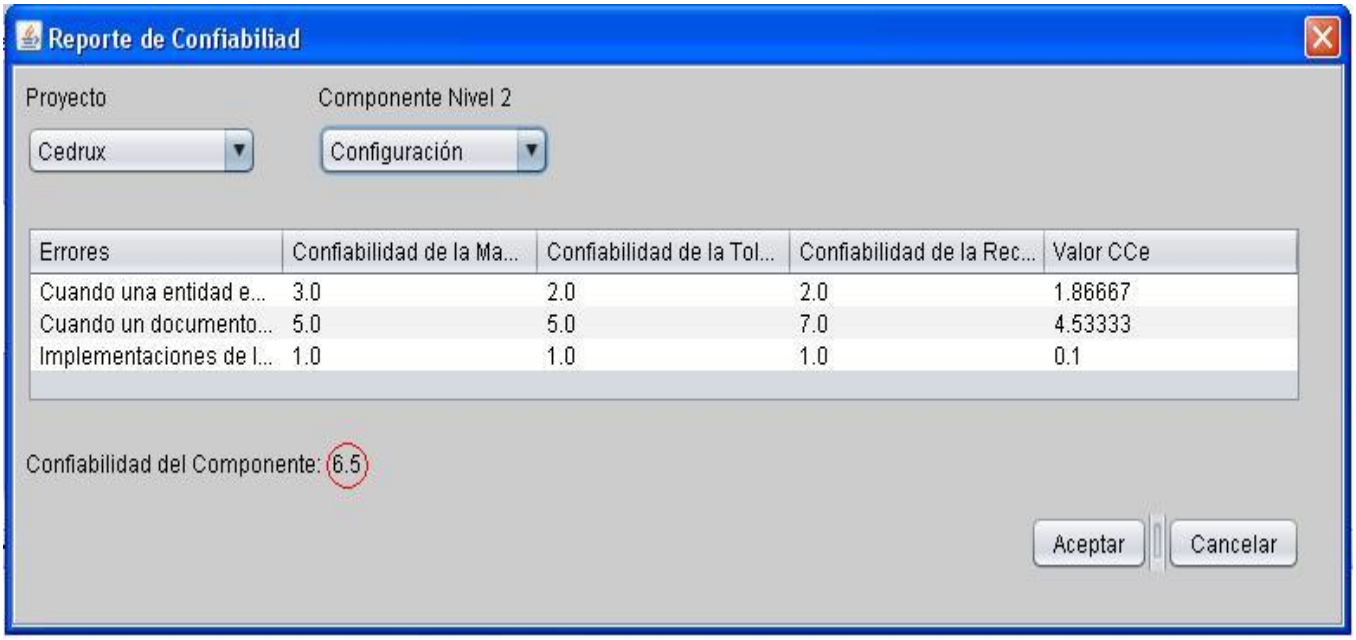

Figura 11. Cálculo de la confiabilidad para los subsistemas Configuración

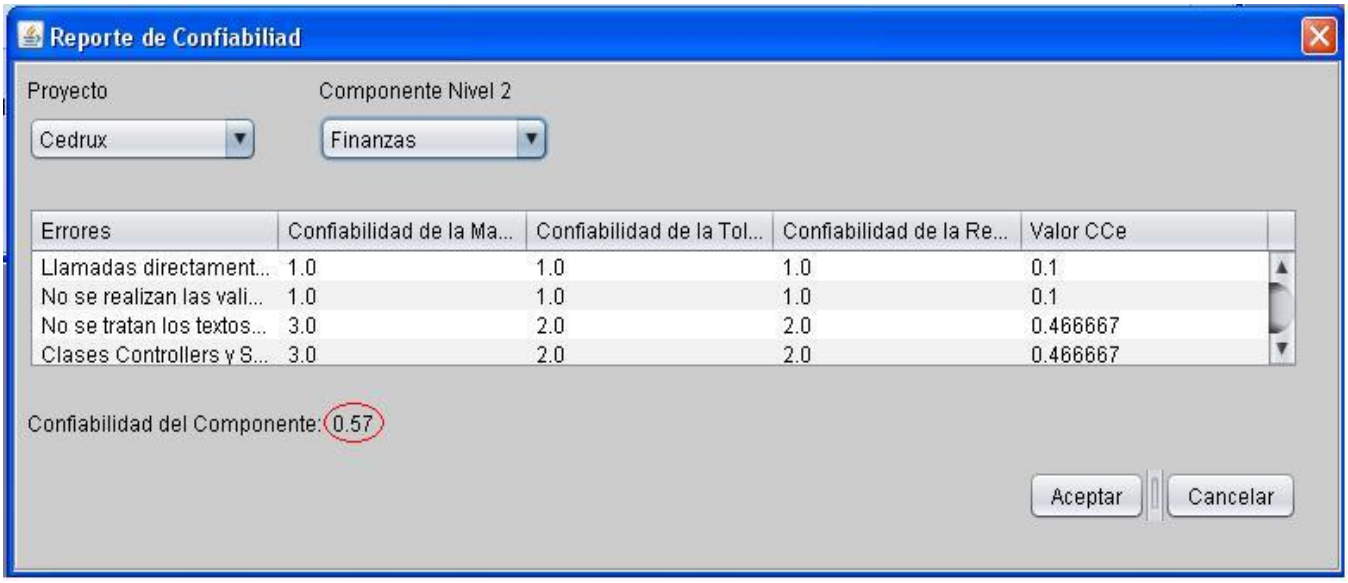

Figura 12. Cálculo de la confiabilidad para los subsistemas Finanzas

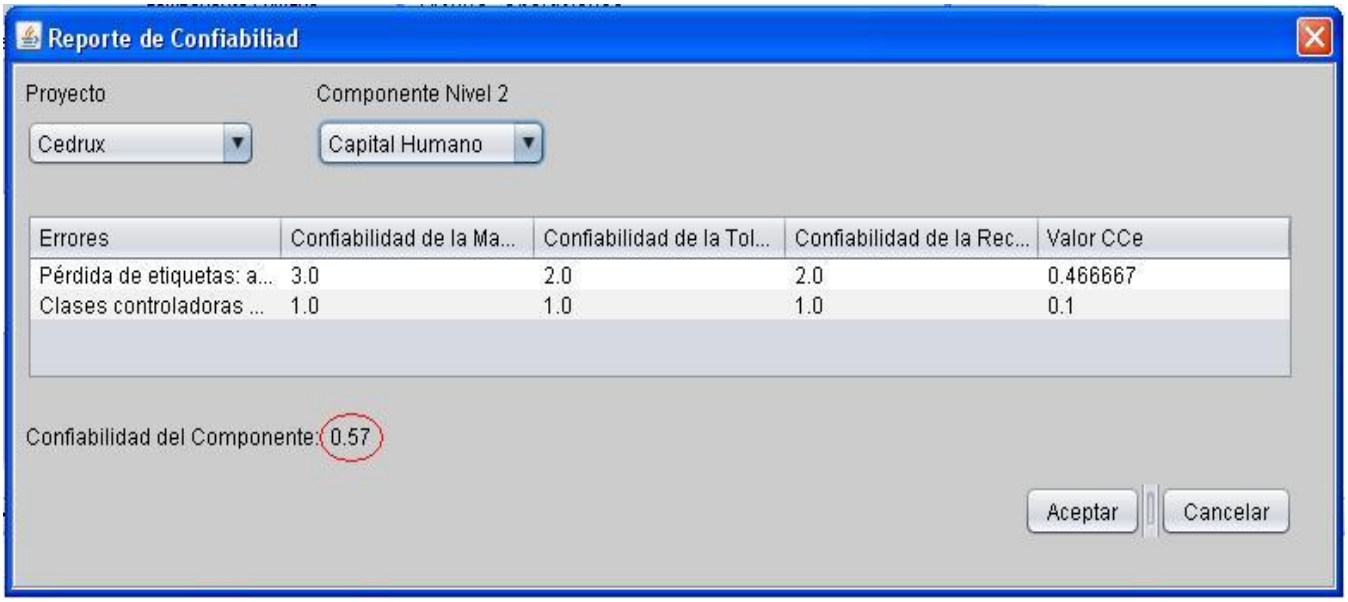

Figura 13. Cálculo de la confiabilidad para los subsistemas Capital Humano

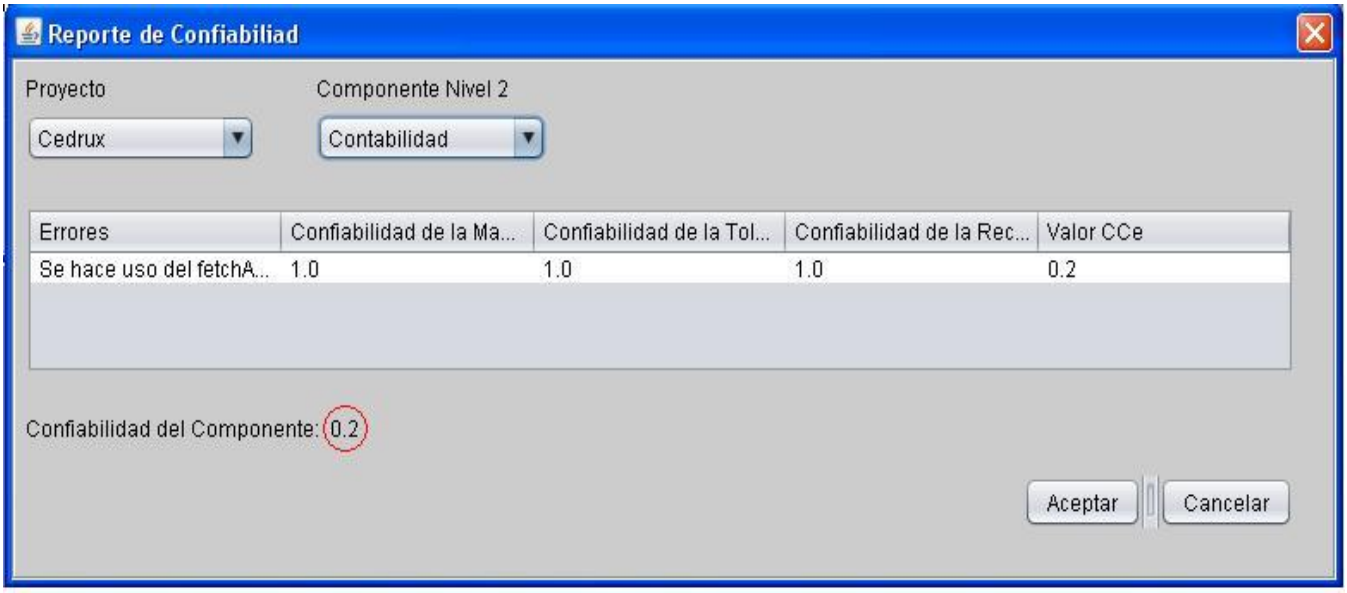

Figura 14. Cálculo de la confiabilidad para los subsistemas Contabilidad

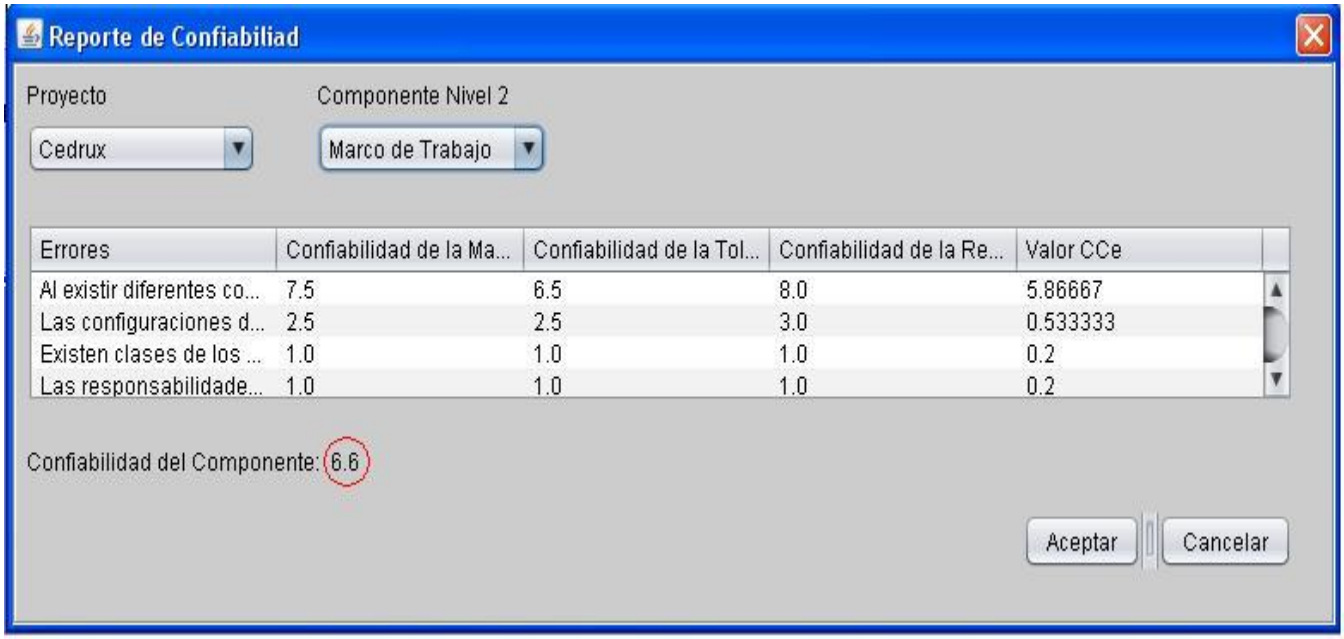

Figura 15. Cálculo de la confiabilidad para los subsistemas Marco de Trabajo

3.4 Análisis de los resultados

La puesta en práctica de la herramienta, con los datos de la muestra, arrojó valores de confiabilidad para el error y el componente. Los valores obtenidos (valor actual), los esperados y la diferencia entre estos se muestran en las tablas 21 y 22. Se observa en la tabla 21 que en el 89 % de la muestra la diferencia arroja valores por debajo de 0.10 por lo cual se consideran valores exactos.

Componentes	Valor esperado confiabilidad del componente	Valor actual confiabilidad del componente	Diferencia
Inventario	0.83	0.83	0
Facturación	0.83	0.83	0
Activos Fijos	0.83	0.83	0
Finanzas	0.57	0.57	0
Auditoría	0.53	0.57	0.04
Configuración	6.5	6.5	0
Contabilidad	0.20	0.20	0
Capital Humano	0.57	0.57	0
Marco de Trabajo	6.46	6.6	0.14

Tabla 21. Comparación entre los valores esperados y los valores actuales

Se observa en la tabla 22 que en el 96 % de la muestra la diferencia arroja valores por debajo de 0.10 por lo cual se consideran valores exactos.

Componentes	Valor esperado confiabilidad del error	Valor actual confiabilidad del error	Diferencia
Inventario	0.10	0.10	0
	0.10	0.10	0
	0.20	0.20	0
	0.53	0.533333	0.003333
Activos Fijos	0.10	0.10	0
	0.10	0.10	0
	0.20	0.20	0
	0.53	0.533333	0.003333
Auditoría	0.10	0.10	0
	0.47	0.466667	0.003333
	0.10	0.10	0

Configuración	1.87	1.866667	0.003333
	4.53	4.533333	0.003333
	0.10	0.10	0
Contabilidad	0.20	0.20	0
Facturación	0.10	0.10	0
	0.20	0.20	0
	0.53	0.533333	0.003333
Capital Humano	0.47	0.466667	0.003333
	0.10	0.10	0
Finanzas	0.10	0.10	0
	0.10	0.10	0
	0.47	0.466667	0.003333
	0.47	0.466667	0.003333
Marco de Trabajo	5.73	5.86667	0.136667
	0.53	0.533333	0.003333
	0.20	0.20	0

	0.20	0.20	0

Tabla 22. Comparación entre los valores esperados y los valores actuales

Aplicando la métrica de precisión A=B/C:

A = 31/33 = 0.94

Se observa que la precisión obtenida es de 0.94 por lo que a mayor cercanía al 1 los datos obtenidos tendrán una mayor precisión en comparación con la realización manual de los cálculos del MECCMA.

3.5 Conclusiones parciales

A lo largo de este capítulo se realizó la validación de la herramienta implementada llegándose a la conclusión de que:

- Con la utilización de la métrica de precisión propuesta por la norma ISO 9126 para la evaluación de la precisión, se validó la herramienta implementada, demostrándose la precisión de los datos obtenidos una vez realizados los distintos cálculos en cada una de las fases del MECCMA. Obteniéndose una herramienta capaz de evaluar la confiabilidad de la arquitectura de Cedrux, la cual ayudara a la hora de la toma de decisiones en la evaluación arquitectónica.

CONCLUSIONES GENERALES

Con la realización del presente trabajo se desarrolló una herramienta para verificar la confiabilidad de la arquitectura del sistema Cedrux, contribuyendo así a la toma de decisiones arquitectónicas por parte de los arquitectos. Es por ello que al finalizar la presente investigación se puede concluir afirmando que:

- Con el estudio del Modelo, las herramientas y tecnologías se logró diseñar e implementar una herramienta capaz de satisfacer las necesidades del cliente y resolver los problemas por los que se decidió comenzar el desarrollo.
- En el transcurso de la implementación de la herramienta se realizaron pruebas unitarias y de aceptación arrojando como resultado un código más depurado proporcionando más facilidad para darle soporte y se le presentó la misma al cliente observándose el grado de conformidad a este último.
- Con la utilización de la métrica de precisión propuesta por la norma ISO 9126 para la evaluación de la precisión, se validó la herramienta implementada, demostrándose la precisión de los datos obtenidos una vez realizados los distintos cálculos en cada una de las fases del MECCMA. Obteniéndose una herramienta capaz de evaluar la confiabilidad de la arquitectura de Cedrux, la cual ayudara a la hora de la toma de decisiones en la evaluación arquitectónica.

RECOMENDACIONES

Luego de la realización de este trabajo se recomienda:

- Integrar esta solución a un entorno de desarrollo arquitectónico donde pueda alimentarse el modelo matemático de simulaciones u otra técnica de construcción o evaluación de arquitectura.

BIBLIOGRAFÍA

ANDINO, A., ENRÍQUEZ, MENESES. 2011. *APRENDIENDO JAVA. Java y sus aplicaciones.* 2011.

Beck, K. 1999. *Extreme Programming Explained. Embrace Change.* s.l. : Pearson Education, 1999.

Bosch, J. 2000. *Design & use of software architectures.* s.l. : Addison-Wesley., 2000.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal. 1996. *Pattern – Oriented Software Architecture. A System of Patterns.* s.l. : John Wiley & Sons., 1996.

Carrascoso, Y. A., Chaviano,Enrique., Céspedes, Anisleydi. 2009. *Procedimiento para la Evaluación de Arquitecturas de Software basadas en Componentes.* La Habana : s.n., 2009.

Codorníu, I. C. L. 2010. *Solución arquitectónica de la Configuración General de Cedrux para la parametrización de negocio del sistema.* Ciudad de la Habana, Cuba : s.n., 2010.

Ecured. 2008. Arquitectura de Software (Evaluación). [En línea] 2008. [Citado el: 17 de 2 de 2013.] http://www.ecured.cu/index.php/Anexo:Arquitectura_de_Software_%28Evaluaci%C3%B3n%29..

—. 2013. Weka. [En línea] 2013. [Citado el: 20 de 4 de 2013.] http://www.ecured.cu/index.php/Weka.

ERIKA CAMACHO, F. C., GABRIEL NUÑEZ. 2004. *ARQUITECTURAS DE SOFTWARE.* 2004.

Fernández, I. O. L. 2011. *Propuesta metodológica para la obtención de los componentes de software en los proyectos del sistema Cedrux. CEIGE.* Ciudad de la Habana, Universidad de las Ciencias Informáticas : s.n., 2011.

Gómez Vieites, D. Á. 2011. *Sistemas de Información Herramientas prácticas para la gestión empresarial.* 2011.

Gómez, O. S. 2007. *Evaluando Arquitecturas de Software. Parte 1. Panorama General.* s.l. : Brainworx S.A, 2007.

IBM. 2013. IBM. [En línea] 2013. [Citado el: 8 de 3 de 2013.] http://www-142.ibm.com/software/products/es/es/category/SWQ50..

ISO, 2001. 2005. *INGENIERÍA DE SOFTWARE—CALIDAD DEL PRODUCTO—PARTE 1: MODELO DE LA CALIDAD (ISO/IEC 9126-1:2001, IDT).* 2005.

J. Stafford, D. R., A. Wolf. 1997. *Chaining: A Software Architecture Dependence Analysis Technique.* Colorado : University of Colorado., 1997.

Bibliografía

José Antonio, Ilizastegui Arriba, D. y. P. R. 2007. *Sistema para la integración continua de proyectos y el control de builds en la empresa Procyon Soluciones.* Ciudad de la Habana : s.n., 2007.

José, H. P., L y Carmen, P. 2003. *Métodologías Ágiles en el Desarrollo de Software.* 2003.

Joskowicz, I. J. 2008. *Reglas y Prácticas en eXtreme Programming.* 2008.

Kazman, R., Clements, P., Klein, M. 2001. *Evaluating Software Architectures. Methods and case studies.* s.l. : Addison Wesley., 2001.

Larman, Craig. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999. ISBN 0-13-748880-7.

MacQueen, J. 1967. *Some methods for classification and analysis of multivariate observations.: Fifth Berkeley Symposium on Math.* California : University of California Press, 1967.

María García Jiménez, A. Á. S. 2007. *Análisis de Datos en WEKA – Pruebas de Selectividad.* s.l. : Universidad Carlos III, 2007.

Martinez, R. 2013. PostgreSQL-es. [En línea] 2013. http://www.postgresql.org.es/node/297.

Milán, Y. 2011. *Propuesta de escenarios arquitectónicos y atributos de calidad para la evaluación arquitectónica del sistema Cedrux.* Ciudad de la Habana : Universidad de las Ciencias Informaticas., 2011.

NetBeans, B. a. 2013. *Bienvenido a NetBeans y www.netbeans.org.* 2013.

Penadés, P. L. y. M. C. 2004. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* s.l. : Universidad Politécnica de Valencia., 2004.

Pérez, K. T. L. 2012. *Modelo matemático para la evaluación cuantitativa de la confiabilidad en la arquitectura de Cedrux.* La Habana, Cuba : Universidad de las Ciencias Informáticas, 2012.

Pérez, M., Grimán, A. and Mendoza, L. 2005. *"AMBIENTE DE EVALUACIÓN DE CALIDAD DE ARQUITECTURAS DE SOFTWARE BASADO EN ECLIPSE.* s.l. : Universidad Simón Bolívar, Dpto. de Procesos y Sistemas, LISI., 2005.

Peter, S. F. H. 2003. *Business Process Management: The Third Wave.* 2003.

PostgreSQL-es.org, R. M. PostgreSQL-es.org. [En línea] http://www.postgresql.org.es/node/297.

Ricart, J. E. 2009. 2009.

Roshandel, R., Banerjee, Somo. ,Cheung, Leslie., Medvidovic, Nenad., Golubchik, Leana. 2006. *Estimating Software Component Reliability by Leveraging Architectural Models.* 2006.

Toro, D. J. C. d. 2007. *Documento de Visión del programa ERP-Cuba. CEIGE.* La Habana, Cuba : Universidad de las Ciencias Informáticas., 2007.

Tuya, J., Ramos Román, Isabel y Dolado Cosín, Javier. 2007. *Técnicas cuantitativas para la gestión en la ingeniería del software.* 2007.

ANEXOS

Anexo 1: Historias de Usuarios

Historia de Usuario	
Código: HU1	**Nombre Historia de Usuario:** Insertar Proyecto.
Modificación de Historia de Usuario Número: Ninguna	
Referencia: RF1	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite insertar un proyecto.	
Observaciones:	

Historia de Usuario	
Código: HU2	**Nombre Historia de Usuario:** Modificar Proyecto.
Modificación de Historia de Usuario Número: Ninguna	
Referencia: RF2	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana

Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite modificar un proyecto.	
Observaciones:	

Historia de Usuario	
Código: HU3	**Nombre Historia de Usuario:** Eliminar Proyecto.
Modificación de Historia de Usuario Número: Ninguna	
Referencia: RF3	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite eliminar un proyecto.	
Observaciones:	

Historia de Usuario	
Código: HU4	**Nombre Historia de Usuario:** Insertar Componente Primario.
Modificación de Historia de Usuario Número: Ninguna	
Referencia: RF4	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera

Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite insertar un componente nivel 2 a un proyecto.	
Observaciones:	

Historia de Usuario	
Código: HU5	**Nombre Historia de Usuario:** Modificar Componente Primario.
Modificación de Historia de Usuario Número: Ninguna	
Referencia: RF5	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite modificar un componente nivel 2 de un proyecto.	
Observaciones:	

Historia de Usuario	
Código: HU6	**Nombre Historia de Usuario:** Eliminar Componente Primario.
Modificación de Historia de Usuario Número: Ninguna	
Referencia: RF6	

Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite eliminar un componente nivel 2 de un proyecto.	
Observaciones:	

Historia de Usuario	
Código: HU7	**Nombre Historia de Usuario:** Insertar Componente Secundario.
Modificación de Historia de Usuario Número: Ninguna	
Referencia: RF7	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite insertar un componente nivel 1 a un componente nivel 2 de un proyecto.	
Observaciones:	

Historia de Usuario	
Código: HU8	**Nombre Historia de Usuario:** Modificar Componente Secundario.
Modificación de Historia de Usuario Número: Ninguna	

Referencia: RF8	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite modificar un componente nivel 1 de un componente nivel 2 que pertenece a un proyecto.	
Observaciones:	

Historia de Usuario	
Código: HU9	**Nombre Historia de Usuario:** Eliminar Componente Secundario.
Modificación de Historia de Usuario Número: Ninguna	
Referencia: RF9	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite eliminar un componente nivel 1 de un componente nivel 2 que pertenece a un proyecto.	
Observaciones:	

Historia de Usuario

Código: HU11	**Nombre Historia de Usuario:** Modificar Error
Modificación de Historia de Usuario Número: Ninguna	
Referencia: RF11	
Programador: Yandy González Cañizarez	**Iteración Asignada:** Primera
Prioridad: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1 Semana
Descripción: La herramienta permite la opción de modificar errores. El sistema debe informar que se realizó satisfactoriamente la operación escogida.	
Observaciones:	

Anexo 2: Tarjetas CRC

Tarjeta CRC	
Clase: Proyecto	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades:	**Colaboraciones:**
Insertar Proyecto Modificar Proyecto Eliminar Proyecto	ComponenteN2

Tarjeta CRC	
Clase: ComponenteN1	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades:	**Colaboraciones:**
Insertar Componente Secundario Modificar Componente Secundario Eliminar Componente Secundario	ComponenteN2 NConfiabilidad Error

Tarjeta CRC	
Clase: ComponenteN2	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades:	**Colaboraciones:**
Insertar Componente Primario Modificar Componente Primario Eliminar Componente Primario	ComponenteN1 NConfiabilidad Error Proyecto

Tarjeta CRC	
Clase: NConfiabilidad	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades:	**Colaboraciones:**
Confiabilidad del Componente	ComponenteN1 ComponenteN2

Tarjeta CRC	
Clase: NConfiabilidadErrorComp	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades:	**Colaboraciones:**
Confiabilidad del error ante el componente	Error

Tarjeta CRC	
Clase: Recorrido	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades:	**Colaboraciones:**
Insertar Recorrido	Error NMadurez NToleranciaFallo NRecuperabilidad NRangoConfiabilidad